

Tabletop Interaction Beyond Touch

Interaction On, Above, Around, and Across Tabletop Surfaces

by

Chi Tai Dang

Dissertation for the academic degree of

Doctor rerum naturalium

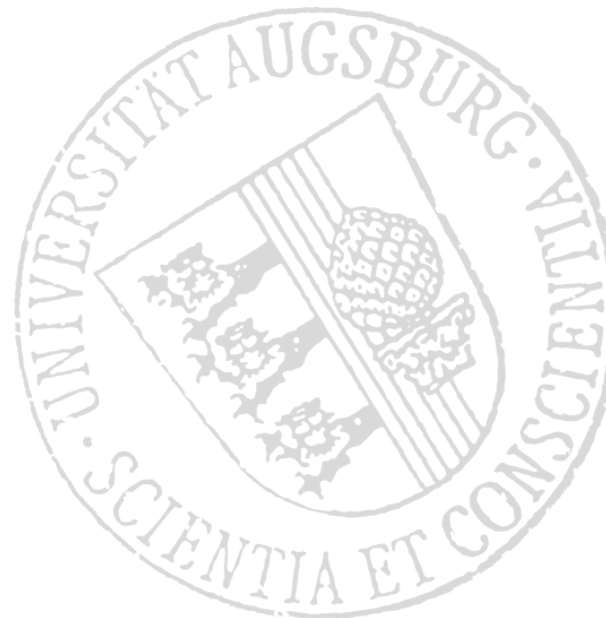
(Dr. rer. nat.)

submitted to the Department of Computer Science

Lab for Human Centered Multimedia

University of Augsburg

2016



Date of examination: 9.12.2016
Supervisor: Prof. Dr. Elisabeth André
Reviewer: Prof. Dr. Bernhard Möller
Reviewer: Prof. Dr. Johannes Schöning



ABSTRACT

INTERACTIVE or digital tabletops are table-like embedded computing systems, which provide a large unified display and input surface. Touch input and tangible artifacts on the tabletop surface represent the most prominent, thus, most often implemented input mechanisms. However, digital tabletops provide much more potential for interaction techniques (e.g., multimodal input or cross-surface interaction). If designers or developers attempt to exploit the full potential of digital tabletops for interactions, they encounter the immediate problem that acquiring an overview of interaction possibilities afforded by digital tabletops can be quite laborious. Without sufficient experiences with digital tabletop technologies, it is further difficult to assess limitations and applicability of different interaction techniques in relation to tabletop technologies. This dissertation addresses this problem by providing and discussing a comprehensible design space for tabletop interaction beyond sole touch input. The design space begins with interaction techniques that happen spatially on a touch sensor surface and extends to interaction techniques that are spatially located above and around a digital tabletop. Finally, the design space considers cross-surface interactions that involve multiple surface-based devices in so-called multi-surface environments.

In each of the areas of the design space, we identified research problems for which this dissertation provides research contributions to advance the field of tabletop interaction. First, we provide algorithms and empirical evaluations for recognition approaches to determine finger orientations and hand distinctions through information extracted from touch contact areas. Second, we present an implementation approach to realize embodied interaction with active actuated tangible artifacts mediated through digital tabletops. And finally, we contribute the design and implementation of a reference multi-surface framework called *Environs* that lay the foundation for enabling cross-surface interactions in multi-surface environments. The framework addresses in particular heterogeneous device ecologies and further enable implementation of low-latency video-based interactive applications, so-called interactive portals, between multiple surface-based devices. In order to foster research and developments of such interactions in multi-surface environments, we provided the reference implementation as open-source software online at <http://www.hcm-lab.de/environs>.

Keywords:

Human-Computer Interaction, Tabletop, Tangible, Multimodal, Design Space, Multi-Touch, Finger-Orientation, Hand-Distinction, Active Tangible, Embodied, Multi-Surface Environment, Multi-Display Environment, Framework

ZUSAMMENFASSUNG

INTERAKTIVE oder digitale Tabletops sind eingebettete Computersysteme in Form von Tischen, welche sich durch eine große einheitliche Darstellungs- und Eingabeoberfläche auszeichnen. Bei diesen Systemen stellen Toucheingaben und Eingaben durch Gegenstände auf der Tabletopoberfläche die bekannteste und daher auch am häufigsten angebotene Eingabemethode dar. Jedoch ist das Potential digitaler Tabletops für die Realisierung von Bedienungstechniken wesentlich größer. Beim Versuch, das volle Potential auszunutzen, werden Designer oder Entwickler allerdings sehr schnell feststellen, dass es sehr aufwändig und mühsam ist, sich einen umfassenden Überblick über die Möglichkeiten digitaler Tabletops zur Realisierung von Bedienungstechniken zu verschaffen. Ohne ausreichende Erfahrungen mit dieser Technologie ist es zudem nicht klar, wie die Anwendbarkeit und mögliche Limitierungen einzelner Bedienungstechniken in Bezug zu den verschiedenen Tabletoptechnologien einzuschätzen ist. Die vorliegende Dissertation nimmt sich dieser Problematik an und adressiert diese durch einen Design-Space für Bedienungsmöglichkeiten, welche über einfache Toucheingaben hinaus geht. Dieser Design-Space umfasst Bedienungstechniken, welche räumlich auf der Tabletopoberfläche stattfinden und erstreckt sich dann über Bedienungstechniken, welche räumlich über oder um den Tabletop herum erfolgen. Abschließend bezieht der Design-Space Bedienungstechniken ein, welche sich über mehrere Geräteoberflächen spannen und daher mehrere Geräte involvieren. Solche Bedienungstechniken finden in sogenannten Multi-Surface Umgebungen statt.

In jedem der Bereiche des Design-Spaces identifizierten wir weiterhin forschungsrelevante Probleme, für welche wir Forschungsbeiträge bereitstellen, um die Forschung im Bereich von Tabletop-Bedienungstechniken voranzubringen. Als Erstes tragen wir Algorithmen und zugehörige empirische Evaluationen zu Erkennungsverfahren bei, um Fingerorientierungen und Handunterscheidungen zu bestimmen, welche lediglich aus Touchkontaktflächen gewonnen werden. Anschließend zeigen wir einen Implementierungsansatz auf, um mittels digitalen Tabletops körperbasierte Interaktionen mit aktiven sich bewegenden greifbaren Gegenständen zu realisieren. Der letzte und größte Beitrag behandelt die Referenz-Implementierung des Multi-Surface Frameworks *Environs*, welches eine Grundlage zur Realisierung von sogenannte Cross-Surface Bedienungstechniken in Multi-Surface Umgebungen bietet. Das Framework unterstützt insbesondere heterogene Gerätelandschaften und ermöglicht die Implementierung von videobasierten interaktiven Anwendungen mit geringer Latenz zwischen mehreren Touchscreengeräten. Um die Forschung und Entwicklung solcher Bedienungstechniken in Multi-Surface Umgebungen zu fördern, wurde *Environs* als Open-Source Software der Öffentlichkeit online bereitgestellt unter <http://www.hcm-lab.de/environs>.

ACKNOWLEDGEMENTS

First of all, my deepest gratitude goes to my supervisor Prof. Dr. Elisabeth André, who gave me the chance to pursue my research ideas and constantly guided me with great knowledge and patience in conducting research. I also want to thank you for your great support during writing this dissertation and transitioning my work from several publications to a single, consistent research thread. Without all the discussions, direction hints, and writing comments, this dissertation would not be possible—thank you!

I also want to thank Prof. Dr. Johannes Schöning for his help in consolidating my research works and the constructive feedback that made this dissertation much more accessible. My gratitude also goes to Prof. Dr. Bernhard Möller for reading the many pages and reviewing this dissertation.

Many students, that I have supervised and worked with, contributed to my research and I want to thank all of you. I want to mention two of the students as it was a pleasure to work with you in courses as well as student works. I want to thank Franziska Mossner for her remarkably great student work in art work, programming, discussions, making jokes, and assistance in studies. I also want to thank Andreas Seiderer for his work on the TabletopCars project who helped a lot to make the project possible.

I am deeply grateful for the support that I received from my family who trusted in me and supported me and made all this possible.

Finally, during the years of my research, I had the chance to travel to great places, meet great people of the research community, and had many fruitful discussions as well as funny talks. Thanks to all the people for that!

CONVENTIONS, LAYOUT, AND EXTERNAL SOURCES

Conventions:

- We use the plural personal pronoun "we" and the plural possessive determiner "our" instead of the single "I" and "my" even if a particular work was solely done by the author.
- Parts of this dissertation, that is, earlier versions, have been published at conferences on human-computer interaction. These parts are designated in the "Contribution Statement" of the particular chapters and sections.
- This dissertation follows the American spelling. For example, it concerns punctuation marks (e.g., comma after *i.e.* or *that is*) or wording (e.g., behavior instead of behaviour).
- Occasionally, this dissertation uses gender-specific names and pronouns to improve readability. Even though, all references to fictional persons are not limited to a particular gender unless explicitly mentioned.
- This dissertation makes use of APA style references to give readers a quick and readable access to the bibliography with the following usage. Citation names that are completely embraced by square brackets represent references to the bibliography. Citation names where only the year of publication is embraced by square brackets represent references to the bibliography and further serve to name the authors in the running text.

Layout:

- Italic font type is used for (1) definitions, (2) textual citations from external work, and (3) to emphasize important words.
- The notes on the margin space give a brief summary of a paragraph or state the question that the paragraph addresses. They are intended to provide a quick overview of the whole structure of a paragraph.
- A small arrow (i.e., ►) prepended to references to chapters, sections, figures, tables, definitions, and equations serves readers to quickly recognize the references.

Sources:

- All figures and tables in this dissertation are the original work of the author with the following exceptions.
- Figures that originate from other authors are indicated by a reference to the related work. Permission to use the figures were granted by the authors.
- The following figures contain cliparts from openclipart.org: ► Figures 3.3a, 3.6, 3.8 and 4.3. Legal usage according to Creative Commons Zero 1.0 License.

TABLE OF CONTENTS

	Page
Abstract	iii
Zusammenfassung	iii
Acknowledgements	v
Conventions, Layout, and External Sources	vi
List of Tables	xii
List of Figures	xiii
1 Introduction	1
1.1 Problem: Focus on Pure Touch Interaction	3
1.2 Research Questions and Contributions	7
1.3 Dissertation Overview	13
2 A Design Space for Tabletop Interaction Beyond Touch	15
2.1 Existing Design Spaces	16
2.2 Three Building Blocks	17
2.3 Chapter Summary	18
3 Finger Input Properties of Touch Contacts	19
3.1 Finger Input Properties and Enabling Technologies	21
3.1.1 Overview of Finger Input Properties.	23
3.1.2 Sensing Technologies	25
3.1.2.1 Optical-Based Approaches	25
3.1.2.2 Non-Optical-Based Approaches	35
3.2 Usage and Potential of Input Properties	39
3.2.1 Idiosyncrasies of Touch Interaction	39
3.2.2 Size / Shape of Contact Area	40
3.2.3 Pressure / Pseudo-Pressure / Force	42
3.2.4 Finger posture	44
3.2.5 Remaining Input Properties	44

3.3	Finger Orientation / Direction	45
3.3.1	Usage of Finger Orientation	46
3.3.2	Related Work	49
3.3.3	Diffuse Illumination	50
3.3.4	Determination of Finger Orientation	51
3.3.4.1	A Naive Approach	51
3.3.5	Recognition Rates	57
3.3.5.1	Reference Orientations	59
3.3.5.2	Accuracy of the Reference Orientations	59
3.3.6	Results	60
3.3.7	Discussion	61
3.4	Hand Distinction	62
3.4.1	Usage of Hand Distinction	63
3.4.2	Related Work	65
3.4.3	Center Position and Finger Orientation	66
3.4.4	Concept for Finger Mapping	67
3.4.4.1	Analysis of Finger Arrangements	67
3.4.4.2	Constraints and Conditions	68
3.4.4.3	Temporal Adaptation	72
3.4.4.4	Diversity of Finger Anatomy	72
3.4.5	Empirical Evaluation	72
3.4.5.1	Apparatus	73
3.4.5.2	Experiment	73
3.4.5.3	Results	74
3.4.6	Discussion	77
3.5	Chapter Summary	77
4	Multimodal Tabletop and Tabletop Tangible Interaction	79
4.1	Multimodal Tabletop Interaction	81
4.1.1	Terms and Terminology	81
4.1.1.1	System-Centered Modality	85
4.1.2	Definition of Multimodal Tabletop Interaction	87
4.1.3	Usage of Multimodal Tabletop Interaction	88
4.1.3.1	Speech	89
4.1.3.2	Pen	91
4.1.3.3	Above Surface Gestures / Pointing	92
4.1.3.4	Eye Gaze	96
4.1.3.5	Haptic, Tangible Objects	98
4.1.3.6	Kinesthetic / Proprioception	98
4.1.3.7	Biosignals	101
4.2	Tabletop Tangible Interaction	103

4.2.1	TUI, Tangibles, Tangible Interaction	103
4.2.1.1	TUI	104
4.2.1.2	Tangibles	105
4.2.1.3	Tangible Interaction	107
4.2.2	Categories of Tabletop Tangibles	108
4.2.2.1	Passive Tangibles	109
4.2.2.2	Active Tangibles	109
4.2.2.3	Exogenic and Endogenic Activity	112
4.2.2.4	Active Actuated Tangibles	115
4.2.2.5	Local and/or Remote Controlled	116
4.3	A Study on Multimodal Active Tangible Interaction - TabletopCars	117
4.3.1	Design Rationale: Why R/C cars and why tabletop games?	119
4.3.2	Related Work	121
4.3.3	System Outline	123
4.3.3.1	Microsoft Surface	123
4.3.3.2	Micro R/C Cars	123
4.3.3.3	Physical Obstacles	125
4.3.3.4	Remote Controller Unit	126
4.3.4	TabletopCars Games	128
4.3.4.1	Car Soccer	128
4.3.4.2	Car Crashing	129
4.3.4.3	Fastest Lap	130
4.3.4.4	Parcours	131
4.3.5	User Feedback	132
4.3.6	Embedded and Embodied Interaction	133
4.3.6.1	User Feedback	134
4.3.6.2	Procedure	134
4.3.7	Results and Discussion	135
4.4	Chapter Summary	136
5	Tabletop Centered Multi-Surface Interaction	139
5.1	Definition of Multi-Surface and Multi-Display Environments	143
5.1.1	Multi-Display Environments (MDE)	144
5.1.2	Multi-Surface Environments (MSE)	146
5.1.3	Definitions for MDE and MSE	146
5.1.4	Tabletop Centered MSE	147
5.1.5	Alternative Terms	148
5.2	Usages in Tabletop Centered Multi-Surface Environments	148
5.2.1	Literature Review	148
5.2.2	MSE Design Space	150
5.2.2.1	Input	153

5.2.2.2	Tasks	154
5.2.2.3	Channels	155
5.2.2.4	Base Technologies	157
5.3	Enabling Technologies	159
5.3.1	Sensor and Display Technologies	159
5.3.2	Infrastructure Technologies	160
5.3.2.1	Infrastructure Hardware	160
5.3.2.2	Infrastructure Software	161
5.4	Environs - Interactive Portal - Multi-Surface Environment Framework . .	163
5.4.1	Related Work	165
5.4.1.1	Interactive Portals	166
5.4.1.2	Latency	167
5.4.2	Example Applications	167
5.4.2.1	MediaBrowser	168
5.4.2.2	Public Display Toucher	169
5.4.3	Requirements and Challenges for MSE frameworks	170
5.4.4	Design of Environs	172
5.4.4.1	Application Types	173
5.4.4.2	Application and Framework Layers	174
5.4.4.3	Code Distribution	176
5.4.4.4	Design Considerations for Mobile and Large Surface Computing Environments and Platforms	177
5.4.4.5	Concept of Application Environments	178
5.4.5	Architecture of Environs	181
5.4.5.1	Environs Instances	181
5.4.6	Framework Layers	182
5.4.6.1	Dynamic Loader.	182
5.4.6.2	Core Layer	184
5.4.6.3	Mediator Layer	188
5.4.6.4	Network Layer	190
5.4.6.5	Touch/Sensor Layer	193
5.4.6.6	Interactive Portal Layer for Smart Portal	197
5.4.6.7	Encryption Layer	206
5.4.6.8	Platform-specific API Layer	207
5.4.7	Programming APIs	208
5.4.7.1	Raw API	209
5.4.7.2	Object API	209
5.5	Chapter Summary	216
6	Summary and Conclusions	217
6.1	Design Space for Tabletop Interaction beyond Touch	217

6.2	Finger Input Properties of Touch Contacts	219
6.3	Multimodal and Tangible Interaction	220
6.4	Interaction in Multi-Surface Environments	222
6.5	Future Work	223
Publications of the Author		225
Bibliography		227
Appendix		243
A.1	Source code for Atmel ATmega8 pulse controller.	243
A.2	Schematic Diagram of the Controller-Board for Pulsed Illumination.	245
A.3	Circuit-Board Layout of the Controller for Pulsed Illumination.	246
A.4	Multi-Surface Environments Review-Literature (sorted by index)	247
A.5	Multi-Surface Environments Review-Literature (sorted by authors).	251
Glossary, Terms and Acronyms		255

LIST OF TABLES

	Page
3.1 Classification of tabletop sensing technologies together with an overview of the availability of finger input properties for each technology. DI = Diffuse Illumination, FTIR = Frustrated Total Internal Reflection, DT = MERL DiamondTouch, Depth = Depth-camera based. ✓ = available, d = derivable, +fb = Fiberio [Holz and Baudisch, 2013]	24
3.2 Recognition rates and standard deviation for four methods: 1 = ellipse-method, 2 = ellipse-method + 180-adjust, 3 = Contourtrack + simple slope, 4 = Contourtrack + regression line.	60
3.3 Recognition rates for all participants, the tallest participant, the smallest participant, and the participants' best and the worst recognition rates.	75
4.1 Classification of interaction channels for multimodal tabletop interaction including modality, medium, and code. (I) = Input, (O) = Output, (I/O) = Input + Output.	90
4.2 TabletopCars interactions in the classification of multimodal tabletop interaction.	121
5.1 Classification of MSE interactions	152
5.2 Latencies at 30 frames per second ($\pm 16ms$).	205

LIST OF FIGURES

	Page
1.1 Examples of digital tabletops: (a) An interactive tabletop built as a large rectangle cabinet. Users place everyday objects on the tabletop as with ordinary tables; (b) A MultiTaction Cell tabletop consisting of an input/output surface embedded into a wooden frame. The system resembles an ordinary table.	1
1.2 Examples of digital tabletops: (a) Microsoft PixelSense tabletop; (b) An interactive tabletop at the airport of Madeira.	2
1.3 Structure of the design space and related research questions.	9
1.4 Dissertation overview with keywords indicating each chapter's contents.	13
2.1 The three building blocks – design subspaces – of the design space for tabletop interaction beyond touch.	18
3.1 A physical contact area caused by a bare finger touch on an interactive surface.	19
3.2 (a) An image from an optical sensing surface, which shows a hand and all its fingers that touch the sensor surface (brightness increased for print). (b) Finger input properties that can be determined for finger contact areas.	22
3.3 (a) Basic construction principle of systems of the Depth category with depth camera and sensing surface (Contains cliparts from openclipart.org). (b) A sensor data frame from the depth camera of a system of the Depth category.	26
3.4 The thresholds and regions for touch detection using depth images of the Depth category.	27
3.5 A permanent blind spot in depth images showing mobile displays caused by depth cameras, which are directed perpendicular to the display surface.	27
3.6 Schematic sketch of the FTIR working principle with sensor components (infrared LEDs, acrylic plate, camera) and visualization component (projector) (Contains cliparts from openclipart.org).	28
3.7 Sensor data which shows all fingers of one hand that touch the sensor surface. .	29
3.8 Schematic sketch of the DI working principle with sensor components (infrared sources, acrylic plate, camera) and visualization component (projector). (Contains cliparts from openclipart.org)	30
3.9 Sketch of camera exposure time and infrared source power in DI continuous mode.	32
3.10 Sketch of camera exposure time and infrared source power in DI pulsed mode. .	32
3.11 (a) High power LEDs used for DI pulsed mode. (c) Sketch of controller board for synchronizing infrared camera and infrared sources. (d) Picture of the controller board (b) mounted in the tabletop interior.	33
3.12 (a) A pulsed DI image without diffuser plate. (b) A pulsed DI image with diffuser plate.	34

3.13 Sketch of the contact area on the sensor grid of a capacitance-based sensing surface.	36
3.14 SurfacePoker: A poker game that makes use of hand shape gestures.	41
3.15 Finger orientation in relation to the center position of a finger contact area. . . .	45
3.16 Finger orientation usage examples: (a) A rotary switch with several latches; (b) Adaptation of reported input point for virtual keyboards.	47
3.17 Finger orientation usage examples: (a) Distant object selection using cross-hair through oriented lines; (b) Fingers spanning open angles for area selection. . . .	48
3.18 (a) Diffuse Illumination infrared image and (b) the corresponding contour image.	50
3.19 Finger contact area and its angle to the x-axis.	51
3.21 A Diffuse Illumination raw image with finger contours marked by white pixels.	52
3.20 A small finger contact area that shows a circular shaped contour.	52
3.22 Points distributed on circles around the finger contact positions. Each point spans a path to the corresponding center coordinate of the finger contact.	53
3.23 Finger contour points/locations in a ringbuffer with start and end points (left). List of contour points and its allocation (right).	55
3.24 Example of an elliptical closed finger contour that was detected in a low contrast image. No gate range can be determined.	56
3.25 Finger gestures and combinations used for evaluation: (a) Move an object with the index finger; (b) Move an object with multiple fingers; (c) Grasping with all fingers.	58
3.26 Finger gestures and combinations used for evaluation: (a) Zoom gesture with fingers of one hand, (b) zoom gesture with fingers of both hands.	58
3.27 Annotated index-finger and thumb of the same hand that perform a gesture on the surface. Green/red colored rulers help annotators determine the finger ori- entation.	60
3.28 An example for ambiguous recognition if no hand distinction is available. Ges- tures with one hand (a) and two hands (b) creating the same touch positions and tracks.	64
3.29 Example for conflicts in collaborative tabletop interaction: two different users who perform a pulling away gesture on the same object and at the same time. . .	65
3.30 A finger blob represented as an ellipse with position and angle drawn into a Cartesian coordinate system.	66
3.31 Contact areas of the five fingers of a hand with the intersection points of its back- ward oriented lines (black colored). Distance between center locations (blue colored). An interior angle between oriented lines (red colored).	67
3.32 Variation in distance and angle of thumb to a finger. Combinations are from left with low distance and acute angle to right with far distance and obtuse angle. . .	68
3.33 Oriented lines of two finger contact areas with intersection point in the back together with the distance between two fingers and the normal distance to the intersection point.	69

3.34	Case 2 and 3: Intersection point in front of at least one line; Case 4: Quasi parallel-oriented finger; Case 5: Quasi-reverse oriented finger.	70
3.35	Two examples of the relationship between the distance of two fingers and the normal distance to the intersection point of its backward oriented lines.	71
3.36	False classification with thumb and ring-finger / little-finger. An ellipse depicts a touch contact; the lines indicate the orientation and the numbers denote the recognized handIds.	75
3.37	Examples of conflict cases where one hand is located beneath the other hand. .	76
3.38	Each image consists of two hands with several fingers and their centroids. Centroids provide information to mark the fingers of two different hands.	76
4.1	An ECG sensor attached to a user's leg and a WiiMote controller in the user's pocket.	79
4.2	Physical artifacts on a tabletop surface for tabletop tangible interaction.	79
4.3	Overview of terms and components for multimodal interaction. A refined and updated scheme inspired by Maybury and Wahlster [1998]. (Source of cliparts: openclipart.org).	82
4.4	Multimodal game input using speech and in-air gestures.	91
4.5	A depth camera setup for tabletop interaction above the surface.	94
4.6	A user wearing a mobile eye tracker.	97
4.7	Examples of physical tangible artifacts used in student projects: (a) Real clams; (b) Commercial plastic figures of "Warmachines" as game pieces.	104
4.8	Examples of 3D printed physical tangible artifacts used in student projects: (a) Different colored and shaped tangible battleships; (b) Diverse miniature obstacles as tangible game pieces.	105
4.9	Different types of markers and tangibles. The upper part of the image shows tangibles and the bottom side of tangibles with markers. The lower part of the image shows the vertically mirrored raw image of the sensing surface.	106
4.10	Different types of SLAP-widgets by Weiss et al. [2009].	107
4.11	Categorization of tangible artifacts from a technical point of view.	108
4.12	An example of exogenic active tangible systems through electromagnetic arrays: Madgets by Weiss et al. [2010].	112
4.13	An example of tangibles that provide active tactile feedback to users: Touchbugs by Nowacka et al. [2013].	114
4.14	A test scenario in the lab during development: (a) Depth sensor on the monitor that sense users in front; (b) The remote controller unit connected to the tabletop system; (c) A user performing gestures to control the blue racecar on the tabletop surface.	118
4.15	The microsizers together with according radio remote controls. Each microsizer has a byte tag mounted on the underside.	119
4.16	Tangibles of TabletopCars in the categorization of tangible artifacts.	120
4.17	Outline of the TabletopCars system with controller, sensors, obstacles, and according software components.	123

4.18	The tangible objects used as obstacles in TabletopCars: miniature models of trees, traffic cones, or oil-drums. Each obstacle has a marker glued to the underside.	125
4.19	The front- and backview of the remote controller unit of TabletopCars.	126
4.20	The inner view of the remote controller unit with the microcontroller in the middle and the original remote controls at the left and right.	127
4.21	Car Soccer game: The initial placement of players' cars. Tracked positions are visualized with colored circles.	129
4.22	Car Crashing: The blue car hitting the red car at the side.	129
4.23	Fastest Lap: Two-player mode with both cars on their according track.	130
4.24	Fastest Lap: Single player mode with only one car.	131
4.25	An example course with physical obstacles and virtual oil slicks. Waypoints are placed through touch interaction.	132
4.26	Kinect control modes for TabletopCars: (a) Depth image of a user in the steering wheel metaphor mode; (b) Virtual button-field mode with the car commands. . .	134
5.1	Examples of the variety of nowadays touch-enabled devices. From left to right: smartwatch, smartphone, phablet, mini-tablet, tablet, hybrid laptop.	140
5.2	Example scenarios for multi-surface environment applications. (a) Photo sharing tabletop. (b) Meeting room environment.	141
5.3	Examples for multi-display environments. (a) Conventional displays and projector in a classroom. (b) Multiple LCDs in a lab.	144
5.4	Examples of multi-display / multi-surface environments. (a) Wall-displays in a corridor. (b) Multiple tabletops, tablets, and a notebook display.	145
5.5	Number of relevant MSE papers distributed over publication years.	150
5.6	Number of papers distributed over publication years for the base technologies: MSE framework, Location / Perspective, Personalization, Privacy/Security. . . .	157
5.7	Interactive real-time portals established from a large tabletop surface to multiple small mobile devices that are placed on the tabletop.	164
5.8	Example application: Media Browser running on a large tabletop surface and multiple smaller mobile surfaces.	168
5.9	A Microsoft Surface byte tag attached to the backside of a mobile device. . . .	169
5.10	Example application: Public Display Toucher. Interaction with a large wall-sized display through personal tablet devices.	170
5.11	Layered design of applications that make use of the Environs framework. . . .	173
5.12	Layers of applications using the Environs framework enriched with details and examples.	175
5.13	Multiple application environments in different physical locations distinguished by area / application names. One and the same application environment (e.g., PublicDisplays) may also be spanned across multiple different physical locations.	179

5.14	Device IDs in application environments. Addressing of other application instances happens by means of area name, app name, and device ID.	180
5.15	Architecture of Environs with its layers and components. Multiple instances with each having those layers can be running at the same time.	183
5.16	Steps of asynchronous (A*) and synchronous (S*) API calls.	185
5.17	Mediator network scenarios: (1) Devices that are in the same network. (2) Devices that are in different logical or physical networks.	189
5.18	Network scenarios supported by the Environs framework.	191
5.19	Split of large data for transport over the communication channel. (1) Progress without split. (2) Progress with split.	192
5.20	Gesture recognizer chain of Environs' touch/sensor layer for one sensor data stream.	196
5.21	Orientation-awareness of interactive portals. Portal sources on the tabletop surface may be rotated by any degree.	198
5.22	Portal creation pipeline stages and stage plug-in types of the interactive portal layer.	199
5.23	Capture stage.	201
5.24	Render stage.	201
5.25	Encoder stage.	203
5.26	Portal consumer/visualization stages of the interactive portal layer.	204
5.27	(a) Latency measure of a portal that covers the tablet's physical size on the surface; (b) Latency measure of a high resolution portal source.	205
5.28	Environs Crypt Layer.	206
5.29	The API layer of Environs instances.	207
5.30	Class diagram for the most important objects provided by the object API. . . .	210
5.31	Example cases for platform object states with different lifetimes.	213
5.32	Forward- and backward-propagation of object states by the platform layer. . . .	215

Chapter 1

Introduction

INTERACTIVE or digital tabletops are computing systems primarily characterized by a large horizontal unified input and output surface. Their form factor, size, and height are often reminiscent of ordinary everyday tables such as office desktops, living room tables or kitchen tables (cf. ► Figures 1.1a and 1.1b). Hence, digital tabletops "*rely on users' mental model of traditional*

What are digital tabletops?



(a)



(b)

Figure 1.1: Examples of digital tabletops: (a) An interactive tabletop built as a large rectangle cabinet. Users place everyday objects on the tabletop as with ordinary tables; (b) A MultiTaction Cell tabletop consisting of an input/output surface embedded into a wooden frame. The system resembles an ordinary table.

tables" [Müller-Tomfelde, 2010, p.1] where people, for example, gather around, place everyday objects on the table surface (cf. ► Figure 1.1b), and conduct many "*habitual activities, from work to leisure activity*" [Bellucci et al., 2014]. In comparison to ordinary tables, interactive tabletops extend the possibilities of tables with interaction techniques through electronic interfaces provided by the large horizontal table-surface. For example, digital tabletops afford multi-touch

What is characteristic of tabletop interaction?



Figure 1.2: Examples of digital tabletops: (a) Microsoft PixelSense tabletop; (b) An interactive tabletop at the airport of Madeira.

input through human fingers on the large display surface. Many tabletop systems also enable interaction by means of physical objects placed atop of the tabletop surface. Although interactive tabletops can be used in single-user scenarios, in particular, their table-like form factors and multi-touch input promote them for simultaneous user input of multiple users, or collaborative usage scenarios in which groups of users work face-to-face around the digital tabletop [Rogers and Lindley, 2004].

Historical
development of
tabletop research.

HISTORICAL DEVELOPMENTS. The first developments of research works related to digital tabletops surfaced more than 25 years ago [Buxton, 2007, Bellucci et al., 2014]. As early as 1985, pioneer works with ideas and concepts of digital tabletops have already been presented in the system "VideoDesk" by Krueger et al. [1985] or the system "DigitalDesk" by Wellner [1993]. Since then, researchers constantly developed tabletop prototypes and studied the phenomena around interaction with digital tabletops [Müller-Tomfelde, 2010, p.3-10]. During the past decade, a large research community has gathered around tabletop research. Many advances in terms of implementation approaches (e.g., [Han, 2005]) and declining prices for components (e.g., short-throw projectors) have accounted for this development. Those advances enabled easier construction of digital tabletops at low prices. As a result, digital tabletops have become widespread in the research and the DIY¹ community. Another reason for the development of the community is certainly the research conference ITS² that started in 2006. This conference was particularly dedicated to digital tabletops and contributed a lot of the research results.

There is a large
research community
around tabletops.

¹ Do-It-Yourself

² International Conference on Interactive Tabletops and Surfaces

COMMERCIAL DEVELOPMENTS. The emerging availability of commercial interactive tabletops, such as the Microsoft Surface tabletops (cf. ► Figure 1.2a), MultiTaction Cells³ (cf. ► Figure 1.1b), or Ideum tables⁴, fostered implementation of interactive digital tabletops at public places such as showrooms, exhibitions, airports (cf. ► Figure 1.2b), or museums. Although dissemination of interactive digital tabletops at public places is constantly increasing, the absolute number is still limited and most people are more likely to see interactive digital tabletops in cinema movies or TV shows.

Digital tabletops in the public.

TABLETOPS OR SURFACES. There are several terms that have become common in the research community, such as tabletop interaction, surface computing, tabletop computing, or surface interaction. Bellucci et al. [2014] discussed the notions of these different terms, outlining that the term "surface" also includes research about vertically mounted surfaces (e.g., wall-mounted displays) or small mobile surfaces (e.g., smartwatches, smartphones, tablets). In this dissertation, we draw upon the commonly used terms and follow the distinction in [Bellucci et al., 2014], that is, we make use of the term "tabletop" if only large (horizontal) table-sized devices are meant. If we also include large vertical touchscreens or small mobile touchscreen devices, then we make use of the term "surface".

Different terms in tabletop research works.

Usage in this dissertation.

1.1 Problem: Focus on Pure Touch Interaction

Interaction techniques provided by digital tabletop applications, for example, as showcased in social media, video-portal platforms, Hollywood movies, TV shows, or at public places (e.g., information points at airports, cf. ► Figure 1.2b), are often merely based on touch input. Certainly, one of the reasons that have a strong influence on this design choice is rooted in the software packages shipped with digital tabletop systems, for example, platform SDKs, stock applications, or examples of applications and templates. They prominently show and teach how to build applications that solely rely on the x/y-coordinates of touch input. As an example, the following excerpt of publicly available SDKs (collected in Nov., in 2015) shows that substantial portions (~83.6%) of the included code examples only make use of the x/y-coordinates of touch input.

Problem: Tabletop interaction often merely allow touch input.

Tabletop platform SDKs often focus on touch input.

- 100.0% Microsoft Surface SDK1 App Suite (3 of 3)
- ~66.6% Microsoft Surface SDK1 and Samples (10 of 15)
- ~61.5% Microsoft Surface SDK2 and Samples (8 of 13)
- ~95.2% MultiTaction Cornerstone SDK and stock Apps (40 of 42)

³ <http://www.multitaction.com>

⁴ <http://ideum.com/touch-tables>

The remaining input types – utilized in the SDK samples – are based on the detection of visual markers placed on the tabletop surface. This is not surprising as touch input is the most prominent and obvious input type afforded by digital tabletops. In the following, we abbreviate interactions based on only the x/y-coordinates of touch input with the words "sole/only touch" or just "touch input".

Inexperienced designers and developers.

Another reason for the dominance of touch input is that digital tabletops are (still) not widespread, as we have outlined before. Thus, the experience of designers, developers, and software engineers with this technology is often limited and – as a consequence – interaction techniques tend to be designed on the basis of known technologies such as smartphones or tablets. Of course, digital tabletops could be interacted with like "very large smartphones or tablets", but this interaction scheme falls short in exploiting the potential of digital tabletops (e.g., interaction across surfaces or multimodal input based on speech, pen, biosignal, eye gaze, or hand, body, head gestures).

Limitation to touch due to compatibility or restrictions.

The limitation to sole touch input might also be caused by technology-related considerations and restrictions, such as idiosyncrasies of different enabling technologies for the realization of interaction techniques, which often widely vary in hardware capabilities as ascertained by Scott and Carpendale [2006]. A consequent and logical design decision due to this circumstance is to restrain interaction techniques to touch input, which is commonly available (on all platforms).

Users tend to use single touch only.

Furthermore, users tend to use single touch with one finger even though multi-touch input or gestures are supported as reported by Wigdor et al. [2007] or Schöning et al. [2009]. According to the review of Bellucci et al. [2014] on horizontal interactive surfaces, the reason for this behavior is two-fold. First, users follow the typical click-and-point interaction that they learned for desktop computing environments. Most users are familiar with this kind of interaction due to the prevailing usage of desktop computing with mice and keyboards. Second, after users (in the studies) were told about more interaction possibilities that go beyond sole touch on one single point, they started to explore new interaction mechanisms. Both can be summarized as lack of knowledge about interaction techniques that are possible with digital tabletops.

Lack of knowledge.

There is much more potential for interaction beyond sole touch input.

INTERACTION POTENTIAL OF TABLETOPS BEYOND TOUCH. Digital tabletops offer substantially more potential and possibilities for realizing interaction techniques that go beyond sole touch input and marker detection. If only touch input would be provided by tabletop applications or rather other means of interaction would be neglected, then – with the increasing proliferation of digital tabletops and applications – more and more potential users would be trimmed to only rely on touch input.

Such a development would create a quite unbalanced awareness of interaction techniques afforded by digital tabletops. Instead, developers and designers should at least try to establish the awareness that digital tabletops offer more means for interaction in addition to touch input or means that complement touch input.

So, how can we foster exploiting the rich possibilities of tabletop interaction? What means are appropriate to support developers, designers, software engineers, and users in this effort? From our point of view, the answer incorporates at least two requirements. First, digital tabletops must implement and provide interaction techniques that go beyond sole touch before such systems are exposed to users. Second, actual users need to know about the interaction possibilities. Usually, users quickly learn from other users or demonstrations, which in turn spreads and reinforces the awareness of commonly known interaction techniques over time. However, for this to happen, the awareness of commonly known interaction techniques for digital tabletops has to be provided and refined by developers and designers. To sum up, the responsibility is at the designer or developer's side.

How to foster the usage of such a potential?

From a designer or developer's view, design decisions can be based on two different preconditions.

- A digital tabletop is already available. In this case, interaction techniques have to be designed based on technological possibilities provided by the platform and technology.
- A digital tabletop has to be acquired. In this case, there are more options for design decisions, as the platform and technology of a digital tabletop can be chosen to enable the intended interaction techniques.

In both cases, designers or developers have to know which platform would enable which interaction technique and vice versa. Here, a resource of knowledge, which shows digital tabletop platforms in relation to their possibilities of tabletop interaction beyond sole touch in a comprehensible and accessible way helps in design decisions. Both cases are addressed in this dissertation by means of a comprehensible design space for tabletop interaction beyond touch.

A design space as a mean for design decisions.

A comprehensible design space not only helps in design and development, but also draws a meaningful overview to oversee and plan research intentions, or to identify research shortcomings and open issues. We will introduce a design space composed of three subspaces for interaction techniques, that is, touch interaction beyond sole touch, multimodal tabletop and tabletop tangible interaction, and tabletop interaction in multi-surface environments.

Research problems identified through the space design space.

Further research problems.

In each of the subspaces, we identified research shortcomings that we consider as research problems of high importance. These further research problems will be introduced in the remainder of this section. In addition to providing a comprehensible design space, this dissertation addresses these identified research problems with dedicated contributions.

Problem: Finger orientation and hand distinction are neglected for tabletop interaction.

TOUCH INTERACTION BEYOND TOUCH. Touch input provides more possibilities (i.e., finger input properties) for interaction techniques than merely x/y-coordinates, as we will extensively discuss in ► Chapter 3. Across all input properties, the direction of a finger while touching a sensor surface - the so-called finger orientation – has rarely been considered for interaction as also confirmed by Wang and Ren [2009]. To improve this situation, Wang et al. [2009] presented an approach to determine the finger orientation for a certain kind of digital tabletop technologies called FTIR (Frustrated Total Internal Reflection, cf. ► Subsection 3.1.2.1) . However, the availability of the finger orientation in other tabletop sensing technologies remained incomplete. For instance, DI-based (Diffuse Illumination) technologies do not provide the benefits of FTIR-based technologies, which make FTIR-based recognition approaches unsuitable (or at least not optimal) for those technologies. Though, a higher availability of approaches for recognition of the finger orientation across all sensing technologies would foster employing the finger orientation in interaction techniques on a wide basis.

Availability of finger orientation is incomplete.

Availability of hand distinction.

In addition to the finger orientation, we identified the hand distinction as an under-researched input property, which was motivated by the affordances [Gibson, 1977] of digital tabletops in terms of collaborative use [Rogers and Lindley, 2004] or in terms of bi-manual interaction and gestures [Moscovich and Hughes, 2008, Benko et al., 2006]. For such usage scenarios, it is quite beneficial to determine whether multiple touch contacts originate from the same hand or from different hands.

Apart from being neglected by research works, recognition approaches for this input property required digital tabletops to be instrumented with auxiliary sensors, such as depth cameras, and to be calibrated to particular environments. An approach that needs only information about touch contacts from a sensor surface would foster the usage of this finger input property, as no additional instrumentation of a digital tabletop is required.

Problem: Research addressing embodied interaction with active actuated tangibles are missing.

MULTIMODAL TABLETOP AND TABLETOP TANGIBLE INTERACTION. During recent years, the development and availability of new input devices fruitfully advanced the possibilities for multimodal tabletop interactions. For instance, affordable depth cameras emerged in the game console market, which enabled reliable tracking of user's body for full-body or embodied interaction. At the same time, the rapid development of 3D printers brought affordable and

compact printer devices to the research community, which in turn enabled easy crafting of custom tangible artifacts and so-called active actuated tangibles. The latter is characterized by actuation through small miniaturized engines.

However, active actuated tangibles are under-researched as ascertained, for example, by Bellucci et al. [2014] or Pedersen and Hornbæk [2011]. This is even more the case for research that addresses the combination of both, that is, embodied interaction and active actuated tangibles. Though, the combination of both enables promising and novel tabletop interaction experiences beyond touch, as we will demonstrate in ► Chapter 4.

TABLETOP INTERACTION IN MULTI-SURFACE ENVIRONMENTS. Mobile surface devices, such as smartphones, tablets, or smartwatches, have become one of the most important and personal computing devices in our everyday life during recent years. In combination with digital tabletops, such devices form a so-called MSE (Multi-Surface Environment), which promises for novel interaction techniques and applications that are spanned across multiple surfaces.

Problem: Concepts for MSE frameworks that address heterogeneous device ecologies are missing.

A crucial requirement for establishing MSEs and providing tabletop interaction in such MSEs are enabling technologies in the form of software infrastructure frameworks. All the proposed concepts and implementations of MSE frameworks in the research literature fall short in addressing nowadays heterogeneous surface-based device ecologies and the needs of tabletop interaction. Small mobile surface-based devices are quite different from full-fledged horizontal tabletop surfaces or large wall-mounted vertical surfaces. Such differences must be incorporated into appropriate concepts and architectures for MSE frameworks in order to successfully enable user interfaces or interaction techniques that are spanned across multiple interactive surfaces.

However, to our knowledge, no works have systematically analyzed the requirements and challenges of such heterogeneous MSE frameworks or even presented framework concepts and designs that particularly take those considerations into account.

1.2 Research Questions and Contributions

So far, we have discussed four research problems. In this section, we present the research questions and contributions belonging to the research problems. ► Figure 1.3 sketches the research questions (denoted with RQ) in relation to the problems that we have portrayed. The first and foremost research questions (RQ1) address enabling technologies and possibilities for tabletop interaction that go beyond sole touch input in general.

Research questions
addressing available
means for interaction
beyond touch.

RQ1:

- What enabling technologies and tabletop interaction techniques and approaches beyond touch do exist or have been proposed and evaluated?
- How can we categorize and compare enabling technologies and tabletop interaction techniques and approaches?

One of the goals of this dissertation is to help establish an awareness for the rich interaction potential of digital tabletops. As discussed, it is crucial for researchers, designers, developers, or users to know about technological means for interaction and available or realizable interaction techniques at an adequate level of detail. Potential users are more interested in actual interaction techniques while the other groups require more details and relations of interaction techniques to enabling technologies as well as limitations.

Contribution:
Comprehensible
design space based
on extensive literature
review.

To answer questions of RQ1, we conducted an extensive review of relevant research works that address tabletop interaction beyond sole touch input. By systematically analyzing, sorting, and categorizing the literature, we identified commonalities and criteria to distinguish between different enabling technologies as well as types of interaction approaches and to derive a scheme for classifying them. Based on this scheme, we provide a design space specific to tabletop interaction beyond touch, which includes interaction techniques and related technologies that are sorted into design subspaces as sketched in ► Figure 1.3. The design space serves as an overview and guidance for researchers, designers, developers, software engineers, or users.

Three design
subspaces.

► Figure 1.3 structures the design space into three different design subspaces, which starts with means for interaction that are spatially located on the tabletop surface. That is, we consider the possibilities for interaction that a sole touch contact provides (apart from x/y-coordinates). Next, we move the focus from the tabletop surface to the spatial area above and around the tabletop. The possibilities in this design subspace belong to the topic multimodal and tangible interaction. Finally, the last design subspace emerges if we continue enlarging the spatial interaction area to include multiple surfaces. Such combinations of surfaces establish so-called multi-surface environments, which enable novel and intuitive interaction techniques that are spanned across multiple devices. The interaction techniques and technologies for each of the design subspaces are quite different due to the different characteristics of the interactions and technologies. For example, in "Finger Input Properties of Touch Contacts" we oppose finger input properties to surface sensing technologies, which are constitutive dimensions for works of that category. However, the same dimensions cannot be used for "Multimodal and Tabletop Interaction" because sensor technologies, input

How to structure a
design subspace?

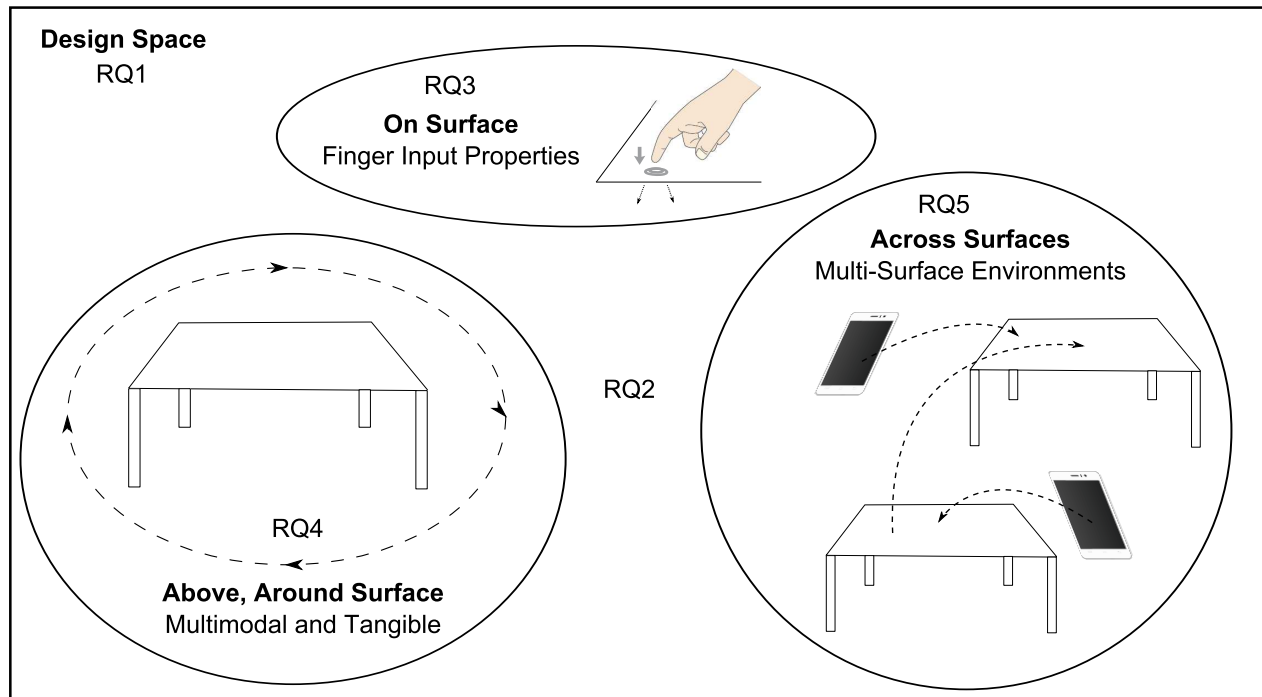


Figure 1.3: Structure of the design space and related research questions.

devices, as well as input types, can be quite different from those in "Finger Input Properties of Touch Contacts". A gyroscope or accelerometer may be used for multimodal interaction but are not suitable to recognize touch contact areas. To build and structure each design subspace, we analyzed the interaction techniques and technologies in each design subspace based on the research questions given in RQ2.

RQ2:

- What are the criteria to distinguish enabling technologies and interaction techniques and approaches?
- What are the limitations, chances, and appropriate usages of them?
- How do enabling technologies relate to interaction techniques in terms of availability and reliability?

Research questions that are common for all design subspaces.

The questions serve to identify criteria that further distinguish technologies, interaction techniques, and their properties (e.g., availability, reliability, certain usages). We present and discuss specific enabling technologies in relation to constitutive input related criteria together with limitations, chances, and usages for interaction at the beginning of each design subspace related chapter. As some input related criteria are limited to certain technologies, we further discuss their availability and reliability in relation to enabling technologies.

Specific criteria for dimensions of the design space.

Paper publication of contributions.

For each of the design subspaces, we have posed further research problems in ► Section 1.1 for which this dissertation provides dedicated contributions. Some of the contributions have been published in the form of peer-reviewed full-papers at high ranked international conferences, such as INTERACT, ITS, TEI, or EICS. We clearly marked them with a dedicated contribution statement at the beginning of a section that contains that contribution. In the following, we show the research questions that drove the problem solving of the research problems in each of the design subspaces.

TOUCH INTERACTION BEYOND TOUCH. As we have outlined, the finger orientation has rarely been used for tabletop interaction and an approach to distinguish finger contacts of multiple hands – without additional instrumentation of tabletops – was even not known. The following research questions address this problem.

RQ3:

- How can we use the finger orientation and hand distinction to realize interaction techniques?
- How can we precisely determine finger orientations or perform hand distinction?

Research questions that address finger orientation and hand distinction.

We discuss the availability and applicability of finger orientation as well as hand distinction for tabletop interaction and present concrete interaction techniques that make use of such input properties.

As no finger orientation recognition approach for Diffused Illumination based digital tabletops was known in the literature, we contribute a robust computer-vision algorithm for this aim together with an evaluation of the precision. The results prove that our approach achieves high precision of finger orientation recognition even for low contrast / low resolution sensor data.

Contribution: Finger orientation algorithm and applications.

There was also no recognition approach for hand distinction known in the literature that do not require additional instrumentation with external sensors (e.g., cameras) and calibration of tabletop systems. Hence, we contribute a heuristic algorithm to distinguish hands for multiple touch contacts only based on two finger input properties, that is, x/y-coordinate and finger orientation. Part of this contribution is an evaluation of the algorithm's precision, which provide results that show high accuracy of the recognition.

Contribution: Hand distinction algorithm and applications.

Both contributions complement and extend the design subspace of available finger input properties for touch interaction techniques beyond sole touch.

MULTIMODAL TABLETOP AND TABLETOP TANGIBLE INTERACTION. The second design subspace related problem and according research questions address implementation approaches to realize embodied interaction together with active actuated tangibles for digital tabletops.

RQ4:

- How can we realize embodied interaction together with active actuated tangibles?
- What are appropriate interaction techniques for that combination?

Research questions that address embodied interaction with active actuated tangibles.

Both topics are under-researched and appropriate interaction techniques with consideration of human factors, such as usability, applicability, or user experience, are scarce in the research literature. Hence, this dissertation contributes an approach to realize multimodal tabletop interaction targeting the combination of active tangible interaction and embodied full-body interaction. We present and discuss the design, architecture, and implementation of the system for software as well as hardware in detail. In particular, we show how to easily enable active actuated tangibles and to control the tangibles through a digital tabletop game with body-gestures sensed by means of a Microsoft Kinect depth camera.

We conducted an empirical study to evaluate the system with two different sets of body-gestures in order to yield results of following kinds. First, we collected experiences expressed by users in terms of acceptance of the interaction approach in general and our concrete prototype implementation. Second, we compared two sets of gestures and provide user preferences and insights into the sets of gestures.

Contribution:
Approach to realize embodied interaction with active actuated tangibles.

This kind of interaction is known from movies such as the "Starwars⁵ compilation" where the actors use their "power" to remotely manipulate real-world objects with hand gestures. Such interaction mechanisms are just fictional and not possible in the real world. However, our system enabled users in the study to perform such kind of interactions and since it was the first time for all the users that they actually did that, we also collected qualitative feedback related to user experience.

This contribution is considered as an early step towards intuitive embodied interaction with self-propulsive real-world objects mediated and remotely manipulated through adaptation of their digital representations by a digital tabletop.

⁵ <http://www.starwars.com>

TABLETOP INTERACTION IN MULTI-SURFACE ENVIRONMENTS. The last design space related problem and according research questions aim at challenges and design concepts for software frameworks specific to multi-surface environments comprised of nowadays heterogeneous surface device ecologies.

RQ5:

- What are the challenges and requirements of multi-surface environment frameworks that aim at heterogeneous device platforms?
- How to design a framework that supports different platforms without implementing, managing and developing the whole framework for each platform separately?
- How to structure and distribute responsibilities for a reasonable and flexible architecture?

Research questions that address multi-surface environments.

Surface-based devices (mobile/static) have changed and are changing rapidly.

As introduced in the according problem statement, device ecologies of mobile surface-based devices and large static devices have changed at a rapid pace in terms of software as well as hardware aspects. In terms of software, there is not only one programming language, framework, or API for application development. Instead, there are many different ones and each of them are mandatory for application development for the particular platform. This is the worst case for software framework concepts and implementations in the existing research literature because many of the proposals are based on concepts of underlying programming languages, such as Java, SmallTalk, or Python, which are at best supported by one platform, thus, far away from real-world cross-platform or multi-platform support. To recap, the world of surface-based devices has changed and probably will change rapidly and so the challenges and requirements on software frameworks for such device ecologies also did change. With such well equipped surface-based devices, visually demanding cross-surface applications and interaction techniques such as interaction through and with real-time video-based applications are not a problem at all anymore from a hardware perspective. However, it is a huge challenge from a software's perspective in particular if many different device platforms are involved.

The last contribution of this dissertation analyzes and discusses the challenges and requirements on such frameworks with the focus on novel and cross-surface interaction mechanisms such as interactive real-time video-based applications. Building upon the identified needs, we propose concepts and an architecture that best support the identified needs. Furthermore, we built a complete working reference implementation, called *Environs*, and made the source code as well as pre-built binaries available to the general public under an open-source license at <http://www.hcm-lab.de/environs>.

Contribution:
Multi-surface environment framework targeting interactive real-time video portals.

1.3 Dissertation Overview

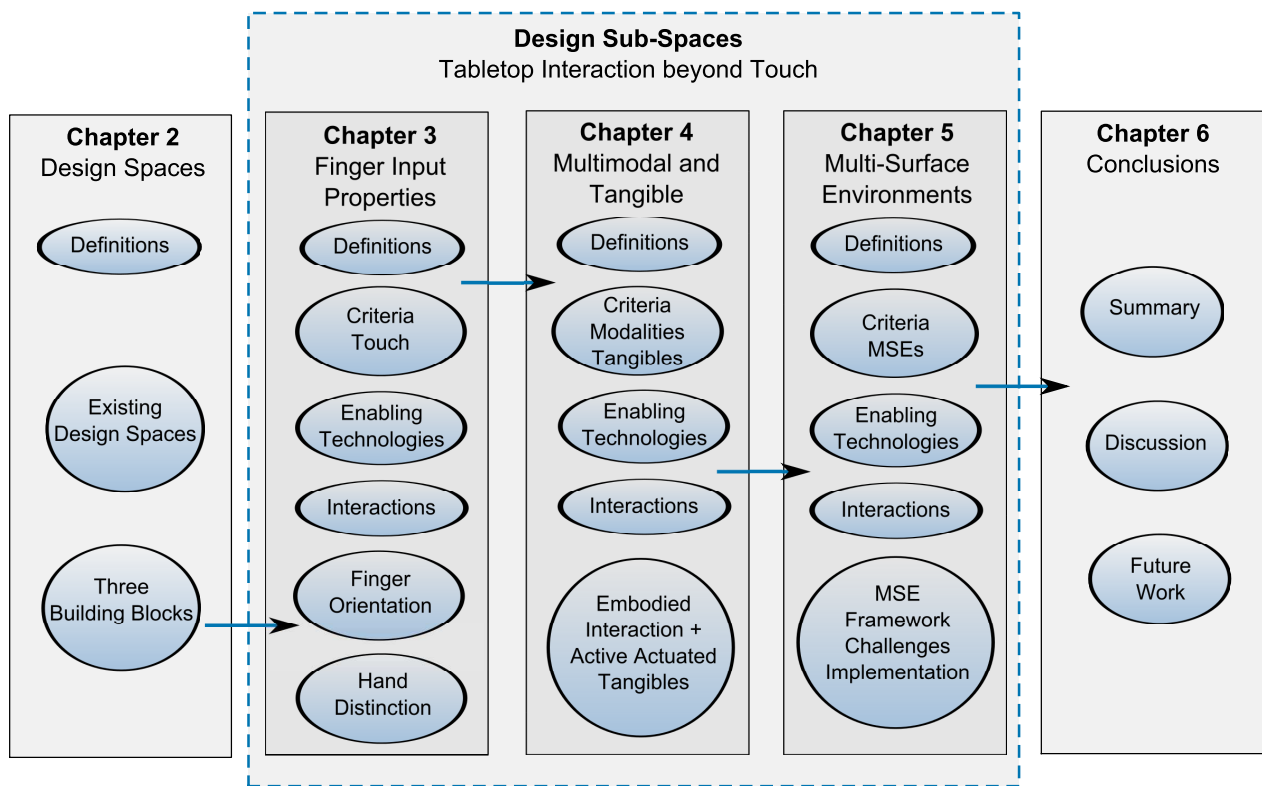


Figure 1.4: Dissertation overview with keywords indicating each chapter's contents.

- **Figure 1.4** gives an overview of the chapters of this dissertation with brief keywords indicating the chapter contents. The chapters are structured according to the problems and research questions discussed in the previous section. Each of the ► Chapters 2 to 5 gives definitions for critical and important terms of the particular chapter topic. These definitions lay the foundation for a common and clear understanding of the content because some terms, such as, for example, modality or surface, are frequently used in a vague, versatile, or even ambiguous manner. Instead of presenting and discussing all related works in a dedicated chapter, we decided to place them in the according chapters and discuss them in the context of the chapters' content. This proceeding creates a better comprehension of the related works' contribution and avoids redundant recurrences of the same content of related works. Where reasonable, we present related works condensed in a dedicated section.
- **Chapter 2** introduces into design spaces and reviews existing design spaces that are related to tabletop interaction beyond touch. The remainder reasons why our design space is broken down to three building blocks, which are discussed in separate chapters.

Structure of this dissertation.

Placement of related works.

What is a design space?.

Design subspace:
Finger input properties
of touch contacts.

► **Chapter 3** investigates the possibilities of interaction techniques beyond touch based on touch contact areas. The chapter presents and discusses a design subspace that oppose finger input properties to technologies employed to sense touch contact areas. Subsequently, The given design subspace is discussed for interaction techniques that have been investigated and evaluated in the existing literature. The remainder of the chapter contributes algorithmic approaches to determine finger orientation and hand distinction as well as applications and interaction techniques that those finger input properties enable.

Design subspace:
Multimodal and
tangible interaction.

► **Chapter 4** is separated into two main sections for multimodal tabletop interaction and tabletop tangible interaction. We discuss related works in order to derive definitions for multimodal interactions as well as different kinds of tangible artifacts, in particular novel active actuated tangibles. Based on the given definitions, the chapter presents and discusses the design subspace that includes tabletop modalities, sensor and tangible technologies, and interaction techniques / approaches. Afterwards, the chapter presents our contribution that combine both, that is, multimodal embodied interaction together with active actuated tangibles.

Design subspace:
Interaction in
multi-surface
environments.

► **Chapter 5** represents that largest chapter of this dissertation and addresses recent and increasing research interest in multi-surface environments and cross-surface interactions and applications. The chapter presents the results of an extensive literature review, which resulted in a design subspace for tabletop interactions in multi-surface environments. Afterwards, key criteria are reasoned in combination with a discussion of the design subspace. The remainder of the chapter presents our reference implementation of the multi-surface environment framework "Environs" in detail with underlying concepts, architecture, and important implementation aspects.

Discussion,
Conclusions, Future
work.

► **Chapter 6** concludes with a discussion of the presented design space and contributions of each chapter. We further show promising future research directions.

► **Appendix A** provides schematic diagrams, circuit board layouts, and a collection of abbreviated references to research works. Those research works were included in the literature review that we conducted for ► Chapter 5 of this dissertation.

A Design Space for Tabletop Interaction Beyond Touch

DESIGN space names a concept that is used to convey and discuss a space of available design choices. As the term design space suggests, the concept originates from design disciplines, such as product or architecture design. It was taken over to the HCI community for communication between designers and researchers, or for discussing design process related issues [Beaudouin-Lafon and Mackay, 2003]. Due to the various disciplines that declared the term design and related tasks for their domains, the term design space may have different meanings. As exemplified by Biskjaer et al. [2014], design space may relate to physical spaces, e.g., design studios or labs, the discourse of design, or as a boundary object in discussions for "*practitioners and researchers to embed their own understandings into the term*". In the HCI context, a design space usually does not relate to physical spaces but rather describes conceptual spaces. In this dissertation, we define a design space similar to the view of MacLean et al. [1991] who described a design space as "*a space of possibilities*" for physical artifacts, whereas we address interaction possibilities and related factors. The following definition describes the perspective that we adopt for the term design space:

What is a design space?

Design space may have different meanings.

Design space is a frame for available design choices or possibilities.

Definition 1. *A design space is an n -dimensional ($n \geq 1$) "conceptional space" [Biskjaer et al., 2014] that spans a frame for available design choices or possibilities for interaction. The design choices vary along the dimensions and result in particular interaction techniques or combinations of those.*

Design space is different from input space.

Bellucci et al. [2014] adopted the term "input space" from Grossman and Wigdor [2007] to frame tabletop interaction and refer the term to "*the physical location where the user can provide input*". They characterized tabletop interaction in a general manner using dimensions such as directness or embodiment. To build their input space, they reviewed and analyzed the input space with strong focus on the input aspect of tabletop interaction. In contrast to Bellucci et al. [2014], we consider tabletop interaction as a bi-directional process where user input and system output has a strong dependency to each other within the whole interaction. For example, active tangible objects that change their shape, form, or location in response to user input have to consider input and output together to adequately characterize according tabletop interactions. Hence, the term design space as defined in ► Definition 1 – which enables a more general view – better suits our aim.

Design spaces serve to classify available interaction as well as discuss interaction choices.

A design space for tabletop interaction help researchers and designers classify and sort certain interaction techniques, get an overview of available alternatives, or discuss them with a clear overview of the possibilities and limitations. We informed and built the design space for tabletop interaction by (1) reviewing and analyzing enabling technologies and interaction techniques for tabletop interaction and (2) by assigning them to categories through distinguishing attributes. The aim of our design space was to show a comprehensible overview of available tabletop interaction beyond touch.

2.1 Existing Design Spaces

This section briefly discusses related design spaces out of the many existing design spaces in the tabletop literature. We included only those works that (1) relate interaction techniques to enabling technologies and that (2) are not limited to a particular application domain or technology.

Classification of tracking approaches and systems.

Kunz et al. presented in [Müller-Tomfelde, 2010, p.51-69] four technical classification schemes, which could be used for deciding on design choices, that is, a classification for tabletop interaction, tabletop tracking approaches, display technology for tabletop systems, and placement of system components in relation to their provided interaction aspects. Their schemes were devised for classification of available tabletop systems from a technical point of view and show the most important interaction types, such as touch, tangible interaction, mouse, stylus, or gestures.

Another classification scheme was presented by Ardito et al. [2015], who summarized the results of a literature review into a scheme with the following dimensions: visualization technology, display setup, interaction modality, ap-

plication purpose, and location. Similar to the schemes of Kunz et al., Ardito et al.'s scheme is technology driven. In addition, they considered several application domains together with types of physical places at which systems may be situated.

Classification of display setups, modality, and applications.

The last design space related work was authored by Grossman and Wigdor [2007] who presented a taxonomy for 3D tabletop systems based on a survey of available systems. The taxonomy is much focused on technologies to realize 3D tabletop, their properties and visualization possibilities as well as input space.

Classification of 3D stereoscopic tabletop interaction.

We are not aware of any design space, which provide a comprehensible overview that address all (currently) available tabletop interaction beyond touch. To fill this gap for research as well as interaction design and system development, one of the contributions of this dissertation is to provide such a design space for tabletop interaction beyond touch that shows interaction techniques along with underlying technologies.

2.2 Three Building Blocks

Due to the vast amount of research related to tabletop interaction, a single all-embracing classification scheme would not allow a clear and easily graspable view for decision making. Hence, we conducted a breakdown of tabletop research works beyond touch and identified three embracing categories into which the available literature can be grouped, that is, finger input properties of touch contacts, multimodal and tangible interaction, and interaction in multi-surface environments. Those categories define the three building blocks in ► Figure 2.1 as a first dimension of the design space.

A breakdown to three building blocks.

The interaction techniques in the building blocks are not exclusive of each other but are considered complementary for interaction design. For example, a particular interaction technique for multi-surface environments may be complemented with pressure sensing, tangible interaction, or include a pen.

In principle, interaction in multi-surface environments may be considered as part of multimodal interaction. Though, from a user interaction point of view, it is sensible to have them separated due to two reasons. First, concrete interaction tasks create quite different user experiences depending on whether the interaction is conducted through one surface or through multiple physically separated surfaces. Second, multi-surface environments enable significantly different interaction techniques, which would not be possible with only one surface, for example, directed bi-manual gestures combining local surfaces for command selection and distant surfaces for target selection.

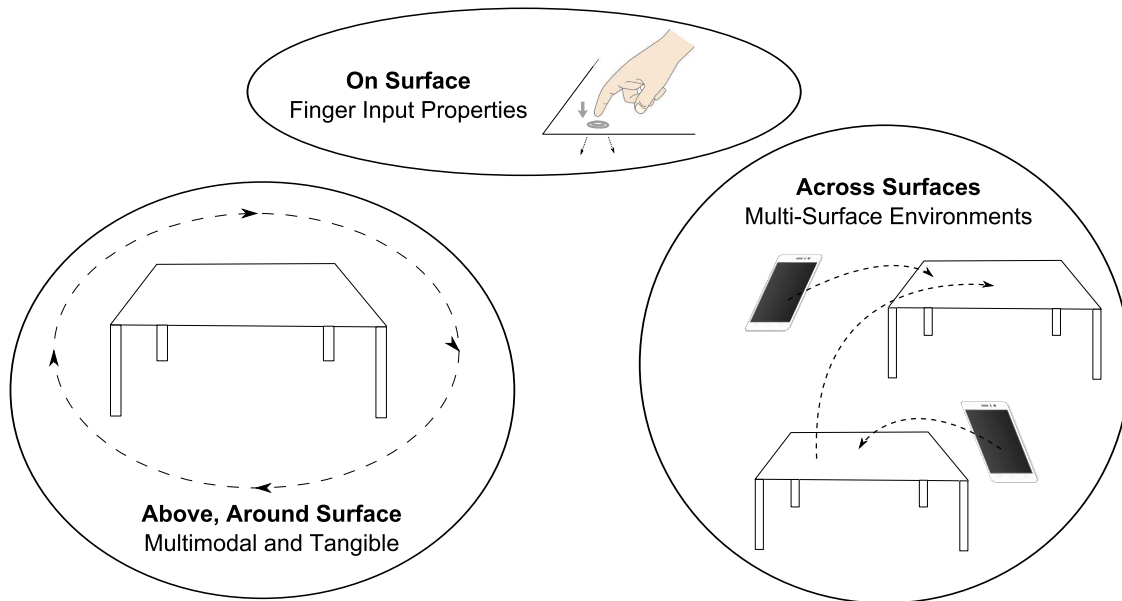


Figure 2.1: The three building blocks – design subspaces – of the design space for tabletop interaction beyond touch.

Each of the building blocks will be discussed in the following with dedicated chapters along with a brief description of key research results of the past decade. Furthermore, each chapter provides classification schemes, which serve to categorize existing interaction as well as for decision making in design processes.

2.3 Chapter Summary

This chapter clarified the term "design space" and its usage in HCI research to establish a clear understanding throughout this dissertation. Afterwards, the chapter discussed existing design spaces from related works that seek to inform tabletop interaction techniques in tandem with enabling technologies. The discussion ascertained that a design space for all (currently) available tabletop interaction beyond touch has not been addressed in the literature in order to motivate the contributions of this dissertation.

We then set out a frame for the design space that will be discussed in detail in the following chapters. This frame surrounds three building blocks – also called design subspaces – that together form the design space for tabletop interaction beyond touch.

Finger Input Properties of Touch Contacts

WE start our exploration of "Finger Input Properties of Touch Contacts" with analyzing the input properties that accompany a bare finger touch on a 2D tabletop sensing surface. Such a bare finger touch creates a physical contact area on the sensing surface that is similar to an ellipse as exemplarily depicted in ► Figure 3.1.

A bare finger touch and its elliptic shaped contact area.

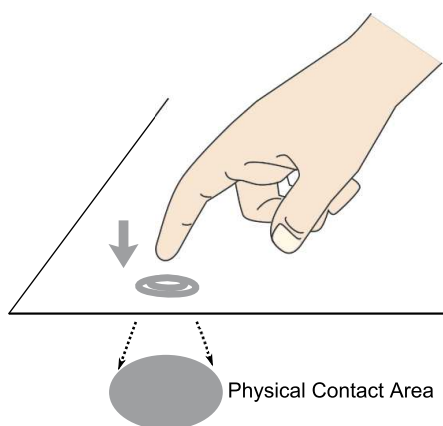


Figure 3.1: A physical contact area caused by a bare finger touch on an interactive surface.

From this contact area, most systems determine the centroid (geometric center point) and derive an x/y-coordinate that is related to the coordinate system of the surface's visual workspace. In addition to the detected information of one single frame of sensor data, sensing systems also derive and manage dynamic touch information. Such information are obtained by keeping track of the detected information from a sequence of sensor data frames over the course of time. Both

kinds of information trigger touch related events in order to supply the current state of detected contact areas (and its information) to user applications, such as movements or changes in size. Applications and application SDKs often make use of so-called touch-down and touch-up events in order to obtain a binary state that serves to simulate a button press. Technically speaking, a touch-down event occurs when the contact area's size increases until it exceeds a certain (touch-down) threshold. A touch-up event occurs when the contact area's size shrinks and drops below a certain (touch-up) threshold or if the contact area (suddenly)

cannot be detected anymore. In practice, the thresholds are platform dependent (e.g., depending on the resolution and size of the sensor surface) and are usually adjusted to avoid false recognition, for example, caused by image noise or infrared-based pens.

Thus far, x/y-coordinates and derived binary states are available on every tabletop system and therefore used by tabletop applications as a common base to realize interaction with users. Both properties are also suitable to provide input compatibility with applications on nowadays prevalent mobile touchscreen devices or traditional WIMP¹ interfaces. For compatibility with WIMP interfaces, the x/y-coordinate is usually treated as a cursor or mouse pointer position while the binary state serves to simulate a mouse button press. Worth mentioning, however, is the issue that traditional WIMP applications are not prepared for multi-touch input. Multi-touch input results in multiple cursors or mouse pointers at the same time. Traditional WIMP applications are usually designed for single point interactions where actions are derived from a sequence of interaction points that have a strong sequential dependency.

Due to the foreseeable large body of available WIMP applications that could be potentially used on interactive tabletops as well as users' familiarity with those applications, researchers have explored and developed techniques to achieve compatibility with traditional mouse input. To name a few examples: Esenther and Ryall [2006] discussed a technique to emulate the hover-state of a mouse pointer, Benko et al. [2006] investigated techniques to perform precise selections with a simulated mouse pointer, and Matejka et al. [2009] presented different approaches to emulate a mouse with three buttons.

While compatibility with traditional mouse input is desirable to enable the easy reuse of legacy applications on interactive tabletops, tabletop applications should not be designed with the aim to emulate desktop computer behavior or to resemble interaction with desktop computers. A large sensing surface, which enables input through human fingers, represents an input device that is completely different from a physical mouse device, which manipulates a graphical mouse pointer. In particular, interaction techniques for interactive tabletops are not limited to techniques based on x/y-coordinates and emulated button presses. Finger contact areas provide more input properties than basic pointer devices such as mice. The contact area depicted in ► Figure 3.1 alone provides more attributes that can be exploited for interaction techniques, such as the lengths of the ellipse axes, or the shape and size of the contact area.

In this chapter, we explore the available finger input properties for interactive tabletops and their availability for different tabletop sensing technologies. Following up, we focus on the finger orientation, that is, the direction in which the

Applications mostly use x/y-coordinates and emulated button presses.

Button presses and x/y-coordinates to interact with traditional WIMP applications.

Research examples of WIMP compatibility.

Touch provides more than x/y-coordinates and emulated button presses.

Exploring finger input properties.

¹ Windows, Icons, Menus, and Pointers

finger is oriented during touch input. So far, this finger input property is rarely considered for interaction techniques, although it provides a rich channel for interaction design that does not require users to learn a new input device. Moreover, users inherently supply the finger orientation with every touch contact. We discuss the interaction related characteristics of finger orientations and examples of how finger orientations can be used for tabletop interaction. Then, we present an efficient approach for detection of finger orientations using simple computer-vision mechanisms. Afterward, we discuss a finger input property called "hand distinction" and present examples of how hand distinction can solve interaction related problems that arise in the collaborative usage of tabletop applications. In order to enable interaction techniques using hand distinction, we provide and discuss an algorithm to derive the hand distinction for touch contacts merely by means of the finger orientation together with the location of a touch contact.

Finger orientation is rarely used. Why?

How can we determine and use the finger orientation for interaction?

How can we determine and use the hand distinction for interaction?

3.1 Finger Input Properties and Enabling Technologies

This section presents an exploration of available finger input properties in relation to different categories of sensing technologies for interactive tabletops. As a basis for the exploration, we first propose a scheme to classify sensing technologies and afterwards provide a classification of available finger input properties using the proposed scheme. The classification also serves as an overview to discuss the reliability of the finger input properties as well as concrete usage examples. The aim is to help developers and designers understand where available finger input properties originate from and what they can be used for in order to enable touch experiences beyond touch. ► Definition 2 establishes a clear understanding of the term **finger input properties**, which we use throughout this thesis.

Classification scheme for sensing technologies and finger input properties.

Overview serves as a design space for discussion.

Definition of finger input properties.

Definition 2. *A finger input property denotes a finger-related value-type whose values can be directly computed or determined using only a single frame of raw sensor data (dataset, image, etc.). If more than one frame of raw sensor data (dataset, image, etc.) is required for the determination, then the property is denoted as a derived finger input property.*

An example of a finger input property that meets ► Definition 2 is the center position of a touch contact. This property can be determined using the centroid of a contact area by means of the contact area from only a single frame of raw

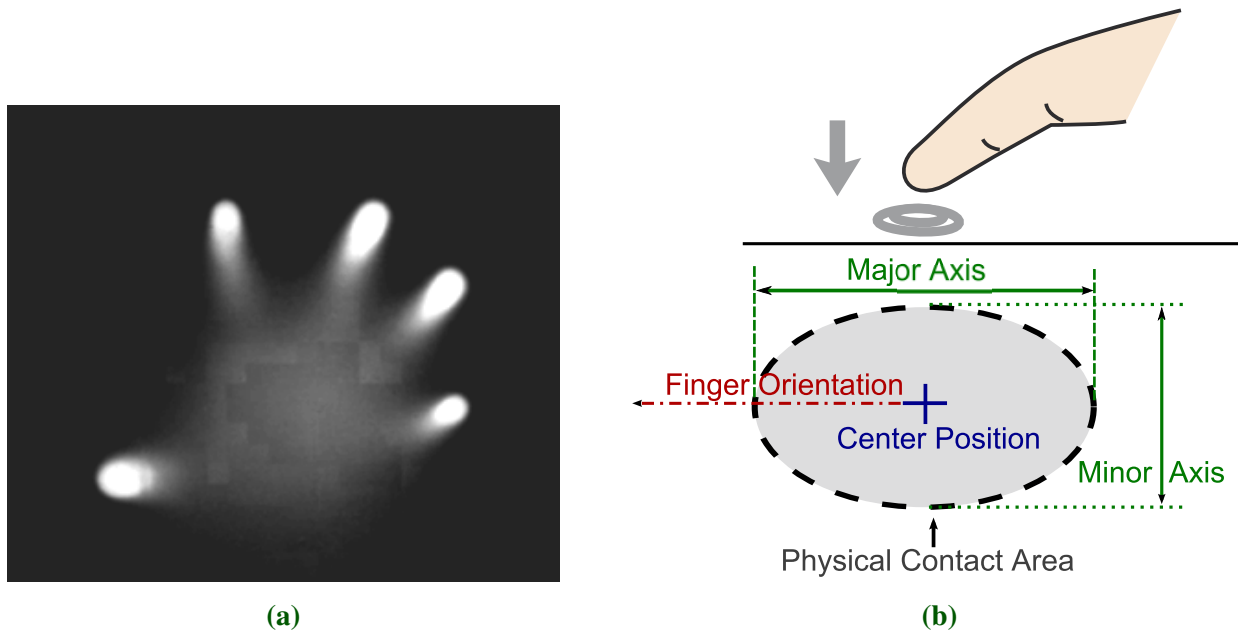


Figure 3.2: (a) An image from an optical sensing surface, which shows a hand and all its fingers that touch the sensor surface (brightness increased for print). (b) Finger input properties that can be determined for finger contact areas.

sensor data. In contrast, a touch-up event can only be derived through observing its corresponding contact areas over time. ► Definition 2 is in line and does not conflict with prior definitions in the existing research literature, for example, by Wang and Ren [2009]. In [Wang and Ren, 2009] and [Wang et al., 2009], the authors distinguish between "finger properties" and "input properties". Finger properties are used for common input related characteristics that stem from fingers or hands, which also include gestures such as tap or flick. The term input property serves for attributing finger properties to categories such as physical or event property [Wang and Ren, 2009].

DERIVED / NOT DERIVED FINGER INPUT PROPERTIES. We further refine our terminology to distinguish between not derived and derived finger input properties such as gestures. From a technical point of view, derived finger input properties open up a broad range of possible properties that all depend on not derived properties. The available number of different finger input properties depends on the actual technology to sense physical touch contacts. Such technologies can be categorized into non-optical and optical-based sensor surfaces. Optical-based approaches generally offer raw sensor data with more information for recognizing properties. To acquire an initial understanding of both categories, let's consider the examples given in ► Figure 3.2. Non-optical-based technologies often enable only finger input properties that are based on the physical contact area as sketched in ► Figure 3.2b. Optical-based technologies perform recognition algorithms on (usually pre-processed) images such as depicted in ► Figure

Finger input properties in the research literature.

Finger properties / Input properties in the research literature.

Difference between derived and not derived properties.

Availability of finger input properties depends on technology.

Optical vs. Non-Optical.

3.2a, which also contain the whole hand of a user with the fingers. The brightest pixels in the image are caused by finger areas that are close to or actually on the sensing surface and are used to determine touch related properties. In addition to properties that are related to touch only, such kind of sensor data also enables the detection of properties that are related to the direction in which the finger points to or related to the finger's pose, for example, whether the finger is held perpendicular or not. Therefore, due to the strong dependency on technologies, we span the design space for "Finger Input Properties of Touch Contacts" along two dimensions: (1) finger input properties and (2) enabling technologies.

Dimensions of finger input properties.

In terms of enabling technologies, Schöning et al. [2008] presented a detailed survey of sensing technologies, which have been established for multi-touch tabletops and have been widely in use such as DI, DSI, FTIR, or capacitance-based sensing surfaces. More recent sensing technologies, for example, technologies that use multi-layer LLP [Takeoka et al., 2010], embedded transducers (e.g., ThinSight [Hodges et al., 2007], PixelSense [Microsoft, 2011]), or depth cameras [Wilson, 2010], can be attributed to one of the approaches described in Schoening's survey with respect to raw sensor data and recognizable finger input properties. For example, raw sensor data (images) sensed by the ThinSight or PixelSense technologies differ from DI images only less. As a result, they enable the recognition of the same finger properties as with the DI approach. The difference among them is mainly reflected in precision and robustness of recognition results.

Finger input properties of more recent technologies.

3.1.1 Overview of Finger Input Properties.

The most often cited work in existing research literature in terms of an overview of finger input properties was contributed by Wang and Ren [2009]. They presented an investigation of finger input properties for which they surveyed a set of available finger properties with the aim to explore the usage of the properties in multi-touch tabletop applications. Their collected set of finger properties was placed into an overview that grouped the properties according to four input aspects: position, motion, physical, and event properties. Wang's overview is useful to show the actual usage of finger properties in the previous research literature. However, their studies and discussions strongly refer to the FTIR sensing technique. Therefore, their overview falls short in giving a comprehensive view that also includes technologies such as DI or non-optical-based technologies.

Available overviews are limited in amount and applicability.

The aim of this chapter is to portray an overview of the availability of finger input properties in relation to different sensing technologies. Such an overview helps developers and designers understand and foresee the applicability of new interaction techniques for tabletop technologies. With this aim in mind, finger in-

Availability of finger input properties in relation to different technology categories.

Properties	Technology					
	Optical-based			Non-Optical-based		
	DI	FTIR	Depth	Capacitance	DT	Resistance
Center position	✓	✓	✓	✓	✓	✓
Size of contact area	✓	✓	d	✓	-	-
Shape of contact area	✓	✓	-	✓	-	-
(Pseudo-) Pressure	d	d	d	d	d	✓
Orientation	✓	✓ or d	✓	d	-	-
Hand Distinction	✓	✓ or d	✓	d	-	-
Hand posture	-	-	✓	-	-	-
User identifier	-	-	-	-	✓	-
Fingerprint	+fb	-	-	-	-	-
Finger posture	-	-	✓	✓	-	-

Table 3.1: Classification of tabletop sensing technologies together with an overview of the availability of finger input properties for each technology.

DI = Diffuse Illumination, FTIR = Frustrated Total Internal Reflection,

DT = MERL DiamondTouch, Depth = Depth-camera based.

✓ = available, d = derivable, +fb = Fiberio [Holz and Baudisch, 2013]

put properties that can only be derived (e.g., motion properties, event properties [Wang and Ren, 2009]) are not technology dependent in the first place. Every technology that enables tracking based on touch-down and touch-up events also enables deriving such properties. Therefore, we omitted those kinds of derived properties from the overview. We took Wang's overview as an initial inspiration and propose a new overview that is more appropriate for the aim of this dissertation.

► Table 3.1 gives an overview of the availability of finger input properties for the most widespread multi-touch tabletop sensing technologies up to date. Each finger input property is represented by a row and each column in the technology area represents a class of sensing technology. The first five properties are finger contact area related properties as sketched in ► Figure 3.2b. All properties listed in ► Table 3.1 can be provided by at least one technology with only one single frame of raw sensor data. In the remaining chapter, we describe the overview in more detail and discuss the potential of the properties for tabletop interaction and applications.

Finger input properties that can only be derived are omitted.

Only one single frame of sensor data for properties.

3.1.2 Sensing Technologies

The second row of ► Table 3.1 denotes a classification of sensing technologies, which includes technologies that (1) have actually been adopted or used in large horizontal tabletop surfaces and that (2) must not require users to carry active or passive auxiliary devices, such as markers, tags, gloves, or a camera.

Second row denotes technology classification.

The classification is structured in a tree hierarchy with a depth of two. The first categorization distinguishes between optical-based and non-optical-based technologies similar to the distinction in Bellucci et al. [2014] and Schöning et al. [2008]. Subsequently, the second distinction categorizes them according to the kind of sensor data that recognition algorithms or tracking processes have to deal with. To denote the categories, the overview makes use of terms that are significant for the underlying technologies (e.g., Capacitance, Resistance) or the term of the earliest approach that produced the particular kind of sensor data (e.g., DI, FTIR).

Technology can be optical-based or non-optical-based.

Technology denoted by indicative terms for sensor data.

The following paragraphs give a brief description of the working principles of the technologies sufficiently to understand the relationship between technology and available finger input properties.

3.1.2.1 Optical-Based Approaches

Optical-based approaches are the most advanced and versatile sensing technologies because of the richness of information that their sensor data provide. Optical-based approaches (most often) rely on infrared light that human eyes cannot recognize. They allow computers to virtually see input properties beyond x/y-coordinates, such as the shape of a finger contact area or the contour of a hand, cf. ► Figure 3.2a. However, such sensor data usually contain unwanted noise, for instance, due to environmental lighting conditions or influences on camera sensors caused by temperature fluctuations. Together with the richness of information, optical-based approaches also pose a bigger challenge to the recognition of input properties in comparison to non-optical-based approaches.

Optical-based approaches provide the most information for finger input recognition.

► Table 3.1 classifies sensor data provided by optical-based approaches into three categories: DI (Diffuse Illumination), FTIR (Frustrated Total Internal Reflection), and Depth. These three categories are sufficient to include all optical-based tabletop sensing approaches that are widely in use due to the idiosyncrasies of the sensor data of each category. Those idiosyncrasies will be discussed in the following and enable recognition systems to use the same image processing steps for all technologies that belong to the same category.

Three categories of sensor data from optical-based approaches.

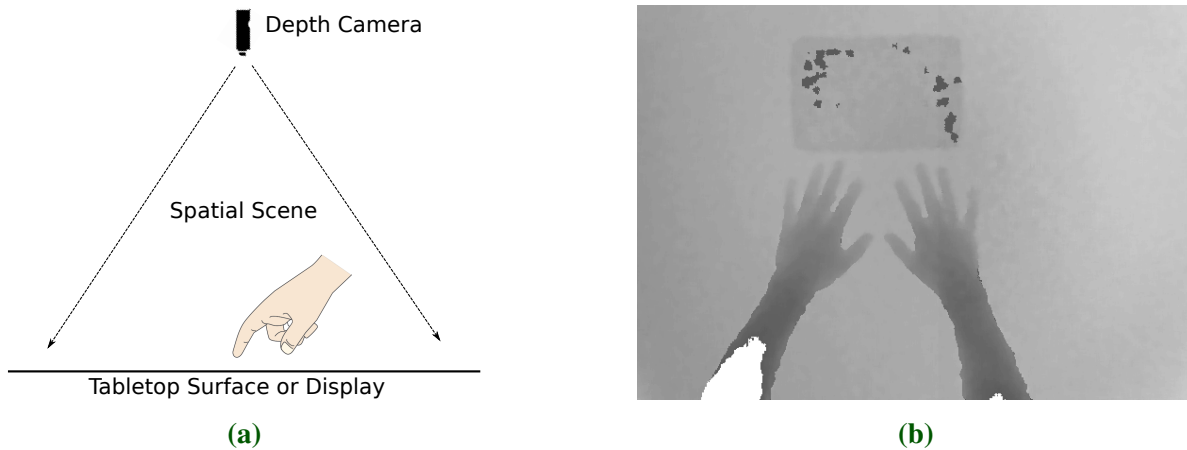


Figure 3.3: (a) Basic construction principle of systems of the Depth category with depth camera and sensing surface (Contains cliparts from openclipart.org). (b) A sensor data frame from the depth camera of a system of the Depth category.

What makes the category Depth unique?

DEPTH. The category Depth summarizes approaches and systems that employ one or more depth cameras as sensor data sources. This kind of sensor data makes such systems unique in this category because of the idiosyncrasies of depth images, in which the values of the pixels reflect the results of depth measurements. In comparison to the other optical approaches (DI, FTIR), depth images require different image processing approaches and pose significantly less reliability, accuracy, and robustness of the recognized finger input properties. However, depth images also provide advantages over the other approaches such as the usage of non-flat surfaces or the usage of the shape of arms or hands for interaction [Wilson, 2010]. In such systems, the camera is usually top-mounted and centered above the tabletop surface as sketched in ► Figure 3.3a. Such a construction enables the camera to capture the whole spatial scene above the tabletop as demonstrated by Wilson [2010]. Depth cameras have become low-priced and widely available since their introduction (late 2010) in the entertainment / consumer market starting with products such as the Microsoft Kinect² or the Asus Xtion³. This development enabled researchers to easily realize tabletop interactions based on depth cameras because the sensing equipment required only a few components, that is, a depth camera and a computer that performs touch recognition.

Common characteristics of Depth-based systems.

Recognition of finger input properties using depth images.

Depth cameras capture images in which the pixel values do not represent colors or intensities but the physical depths in mm/cm between the camera sensor and points on objects in front of the camera sensor. ► Figure 3.3b shows an example of such a depth image with both arms and hands of a user. Those images also do not directly provide the physical finger contact area but enable touch interaction by deriving touch events from the height of fingertips in relation to the

² <https://www.microsoft.com/en-us/kinectforwindows>

³ www.asus.com

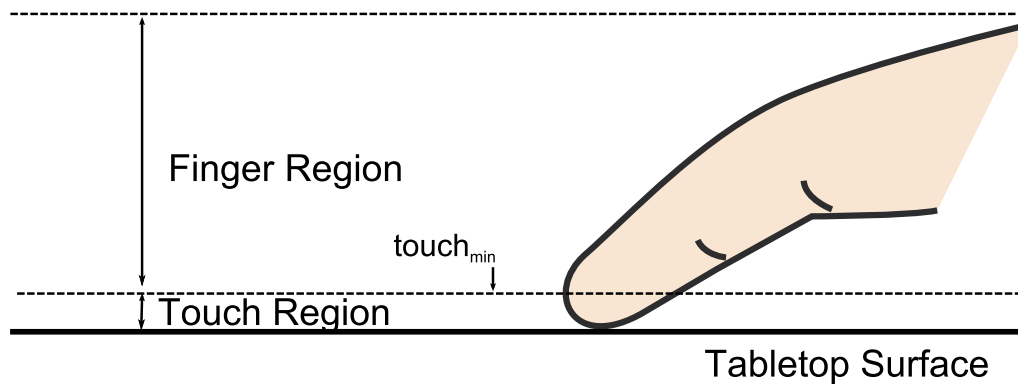


Figure 3.4: The thresholds and regions for touch detection using depth images of the Depth category.

height of the tabletop surface. If this difference in height falls below a predefined threshold, then a physical touch contact is assumed to happen at a location approximated through the observed fingertip area as sketched in ► Figure 3.4. In addition to the fingertip's location, depth images enable detection of more properties (cf. ► Table 3.1), such as the finger orientation or hand posture as demonstrated by Murugappan et al. [2012].

In general, the finger input properties provided by the category Depth are not as accurate and reliable as for the other technology categories of the overview. Furthermore, the camera requires a clear and unobstructed view onto the surface for operation in the first place. For example, if users want to get a closer look on the surface's graphical visualization, their heads may unintentionally cross the camera's view onto the surface and therefore disturb the recognition process. Another limitation that emerges when using traditional (LCD) displays as tabletop surfaces for realizing the output channel shows up as a permanent little blind spot in the center of the surface area as shown in ► Figure 3.5. The research literature also informally refers to this effect as Sauron's Eye [Rädle et al., 2014]. According to Rädle et al. [2014], this effect is caused by the fact that the working

Limitations of
Depth-based systems.

Blind spot in the
center of depth
images.

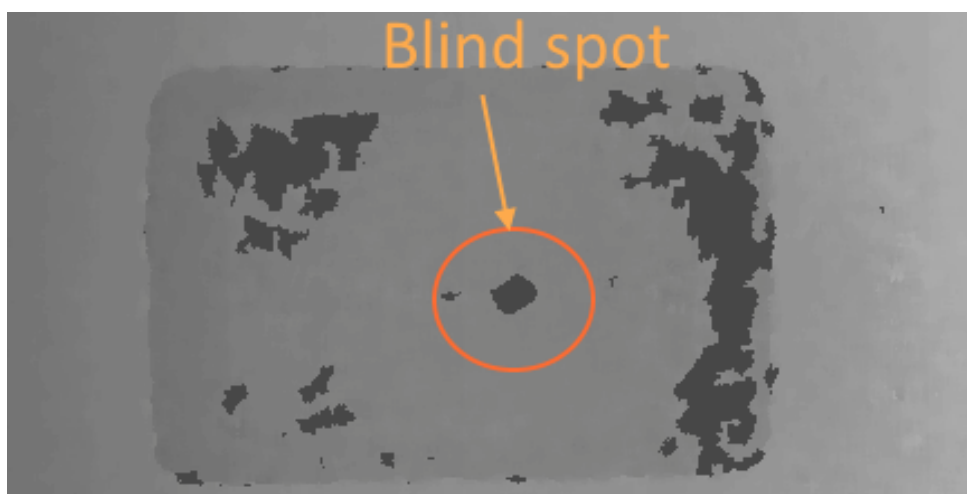
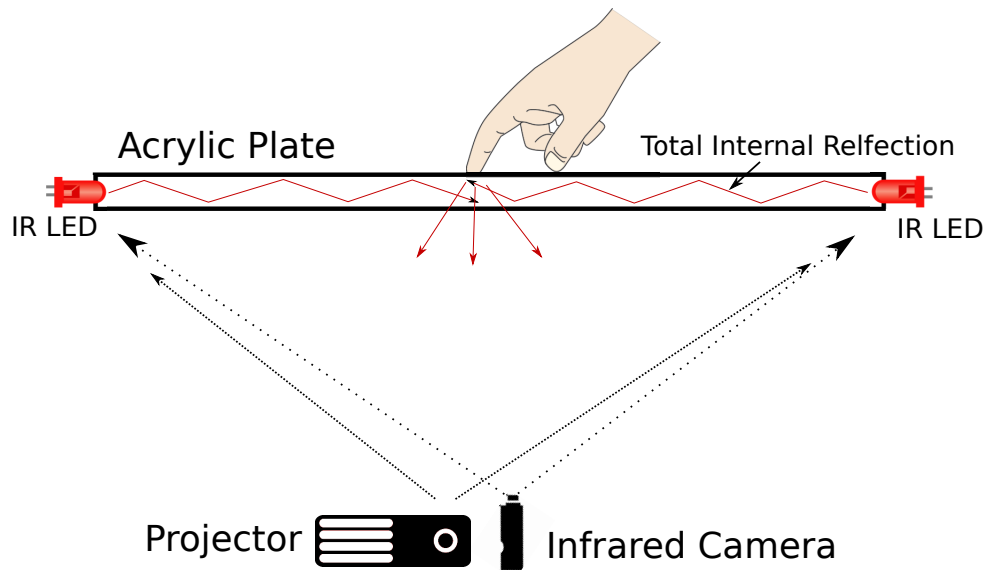


Figure 3.5: A permanent blind spot in depth images showing mobile displays caused by depth cameras, which are directed perpendicular to the display surface.

principle of most depth cameras is based on infrared light. In the center of the surface (90 degrees vertically directed above), too much infrared light is directly

Figure 3.6:

Schematic sketch of the FTIR working principle with sensor components (infrared LEDs, acrylic plate, camera) and visualization component (projector) (Contains cliparts from openclipart.org).



Depth images have potential for tracking of users' heads, arms, or objects.

reflected back from the surface into the depth sensor, which saturates the depth pixels that represent this area. In turn, this circumstance makes the sensor blind in this area. Apart from these limitations, depth images have the potential to enable recognition of interaction related elements that the other technology categories do not enable (without further instrumentation), such as tracking users' heads, arms, or objects that are located further above the tabletop surface.

FTIR tabletop working principle.

FTIR. The term FTIR (Frustrated Total Internal Reflection) tabletop originally refers to systems that make use of a large acrylic plate for the tabletop surface based on the optical phenomenon of total internal reflection. FTIR tabletops have infrared light sources, such as LEDs, mounted on the four sides of the acrylic plate, which emit infrared light into the acrylic plate as sketched in ► Figure 3.6. The different translucent materials, that is, an acrylic plate and air, establish a boundary with different refractive indices. Air has a lower refractive index than acrylic material. The phenomenon of total internal reflection takes effect in this construction when the infrared light within the acrylic material reach the boundary to air (with a lower refractive index). As a result, the infrared light is totally reflected back from the boundary into the acrylic plate. If a human finger touches the acrylic plate, a large amount of the infrared light that was caught in the acrylic plate escapes and gets distracted downwards at the contact location. This fact is exploited by sensing systems to capture high contrast blob images (cf. ► Figure 3.7a) with an infrared camera that is mounted below the acrylic plate.

Origin of the FTIR principle.

The FTIR phenomenon has been known and researched "*since the time of Newton and Fresnel*" [Zhu et al., 1986] and used, for example, for spectroscopy [Harrick and Carlson, 1971]. In 2005, Han [2005] successfully demonstrated the FTIR principle for sensing surfaces in large horizontal tabletops. Since then,

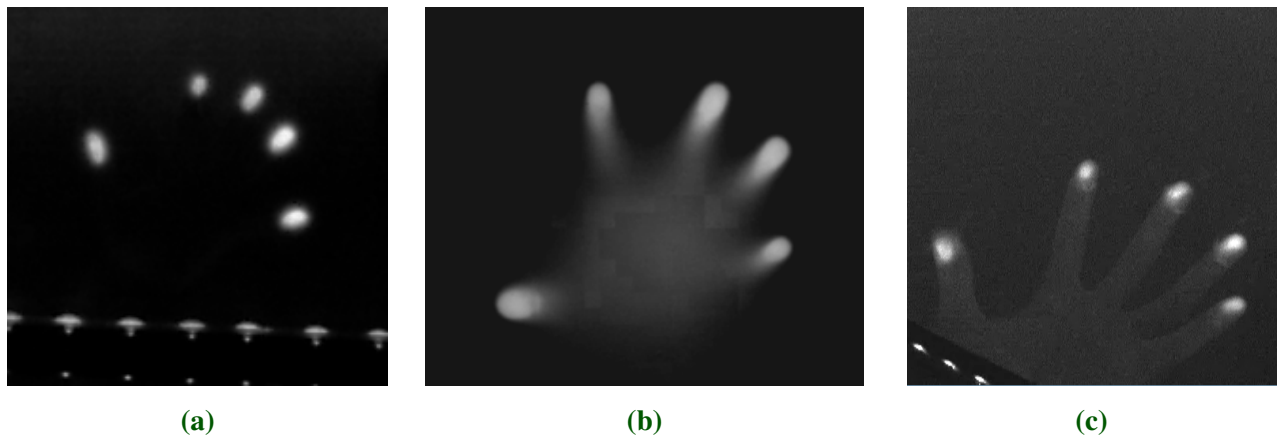


Figure 3.7: Sensor data which shows all fingers of one hand that touch the sensor surface.

FTIR has become quite popular for building interactive tabletops, mainly because such systems are easy to craft at low cost.

The FTIR approach produces high contrast blob images such as depicted in ► Figure 3.7a. Such images are characterized by very bright elliptic shaped spots, so-called blobs, which are caused by physical contact of human fingers with the tabletop surface. Blob images not only originate from technologies employing the FTIR principle but also from other technologies such as FLATIR [Hofer et al., 2009] or FiberBoard [Jackson et al., 2009]. Technologies that have high contrast blob images in common also share the same basic set of recognizable finger input properties. Therefore, the category FTIR consolidates all optical-based approaches that operate on sensor data, which are pre-processed to or resemble high contrast blob images for the recognition of finger input properties.

What makes the category FTIR unique?

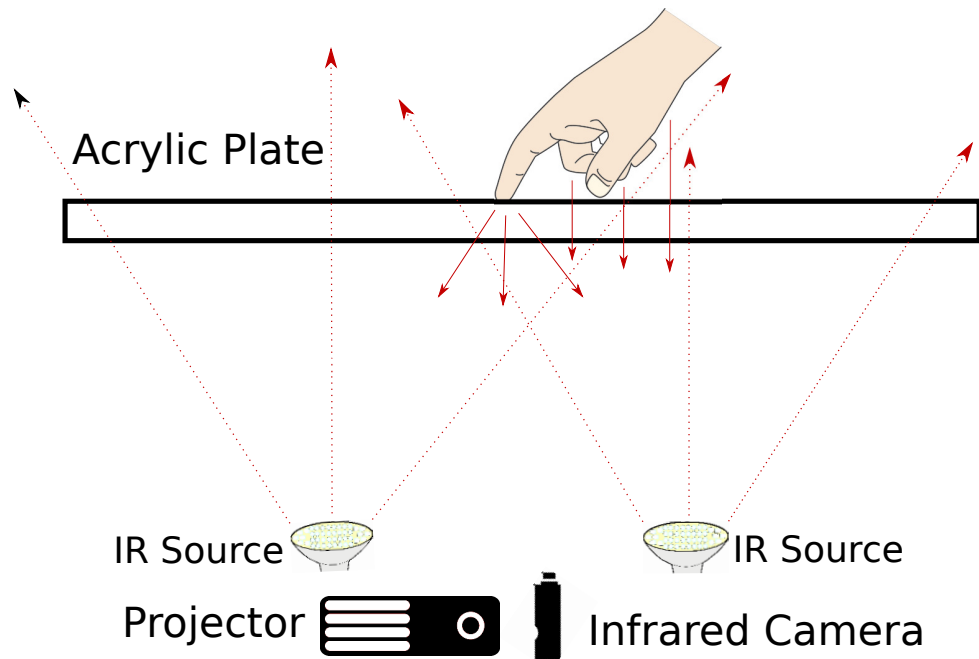
By considering the blob image in ► Figure 3.7a, it becomes clear that determining the center position and the size / shape of the contact area can be easily achieved. The blobs directly correlate to the physical contact areas as discussed and sketched in ► Figure 3.2b. Furthermore, the blobs are usually captured with high contrast, which results in overall high accuracy, reliability, and stability of the recognized finger input properties.

Recognition of finger input properties using FTIR infrared images.

Other finger input properties, such as finger orientation and hand identifier, usually cannot be determined by only a single blob image but have to be derived from a sequence of images [Wang et al., 2009] or a combination of other properties [Dang et al., 2009]. Researchers also have presented more advanced constructions based on the FTIR principle that enable recognition of those properties with only a single frame [Echtler et al., 2008, Iacolina et al., 2011]. Therefore, the overview in ► Table 3.1 indicates the availability of those properties as directly determinable property (extended FTIR) and also as derivable property (classic FTIR).

Limitations of FTIR-based systems.

Figure 3.8:
Schematic sketch
of the DI working
principle with
sensor
components
(infrared sources,
acrylic plate,
camera) and
visualization
component
(projector).
(Contains cliparts
from
openclipart.org)



DI. The category DI (Diffuse Illumination) embraces sensing technologies, which provide sensor data with the richest amount of information for recognition of finger input properties. Hence, such technologies are superior over the other categories. The basic system setup includes a translucent tabletop surface, for example, an acrylic plate and an infrared camera that is mounted below the tabletop surface similar to FTIR tabletop setups. As opposed to FTIR tabletops, the infrared sources are mounted below the tabletop surface (instead of on the side of the acrylic) and emit infrared light directed towards the tabletop surface. By this way, the infrared light sources establish a diffusely illuminated space above the tabletop surface. Physical objects, such as hands, fingers, or arbitrary physical artifacts, on or slightly above the tabletop surface reflect the infrared light back towards the infrared camera. Therefore, when users' fingers and hands approach the tabletop surface, the camera captures infrared images showing the whole hand with its fingers where the brightness of the hands' and fingers' pixels indicate the closeness of the particular "object-pixel" to the tabletop surface.

DI working principle.

Examples of raw sensor data of a DI tabletop are given in ► Figure 3.7b and ► Figure 3.7c. Within such images, not only the physical contact areas are recognizable but also the fingers and palms of the hands. Hence, sensor data of the category DI enable recognition of more finger input properties than the technologies of the other categories. The availability of such information in the sensor data makes the category DI unique over the other optical-based approaches. Therefore, all technologies that are based on such kind of sensor data for finger input recognition are assigned to the category DI in ► Table 3.1. That is, the category also includes technologies such as multi-layer LLP, Bezel-IR, Microsoft

What makes the
category DI unique?

PixelSense, or DSI⁴. Even though some of the technologies are based on a different hardware setup (e.g., laser planes), the recognition steps performed on the images are in principle the same as for the classic DI approach. As a result, the amount of finger input properties that are recognizable using the technologies are also comparable.

From a technical point of view, FTIR sensor data can be deemed as a subset of DI sensor data. Consequently, finger input property recognition on DI sensor data builds on the image processing steps used for FTIR recognition as a first stage and performs additional image processing for further finger input properties (e.g., hand identifier or finger orientation) as a second stage. Depending on whether the second stage image processing requires results from the first stage or not, the stages have to be performed sequentially (required) or can be performed concurrently (not required). If the result of the first stage is required for the second stage, it is appropriate to reuse the recognition result from one sensor data frame earlier as input for the second stage. Usually, the results from one frame to the next one only differ marginally. However, the whole image processing then benefits from parallel processing in order to minimize latency.

Most DI setups tackle with optimization of contrast of camera images, which increase or decrease depending on the amount of light in the environment in which the tabletop is situated. In order to achieve an optimal image contrast independent of environmental light and thus improve finger input property recognition, researchers have presented several extensions to the classic DI setup, which we briefly discuss in the following. The obvious approach is to increase the amount of infrared light that is emitted towards the tabletop surface either by increasing the number of infrared sources or by integrating specialized high power infrared sources or the combination of both. This approach usually entails a proportionally increasing power consumption and results in an increased amount of waste heat produced by power supplies and light sources. In turn, the increased temperature (or fluctuation of temperature) caused by the waste heat negatively influences the camera sensor and increases the noise level within the captured images.

Another promising approach to gain a better contrast of source images is called "Pulsed / Modulated Illumination" and was first proposed by Echtler et al. [2009]. The idea behind this approach addresses the fact that infrared sources in the classic DI setup are continuously emitting light. Though, the involved camera sensor is exposed to the light only a short part of the time, that is, a short period between two frames as sketched in ► Figure 3.9. This fact results in a waste of infrared light energy, which potentially could be used to increase image contrast. Furthermore, many infrared sources can be driven in a so-called "pulsed

Recognition of finger input properties using DI infrared images.

Limitations of DI-based systems.

Adjusting performance by increasing amount of infrared light.

Improving DI by Pulsed Illumination.

⁴ Diffuse Surface Illumination

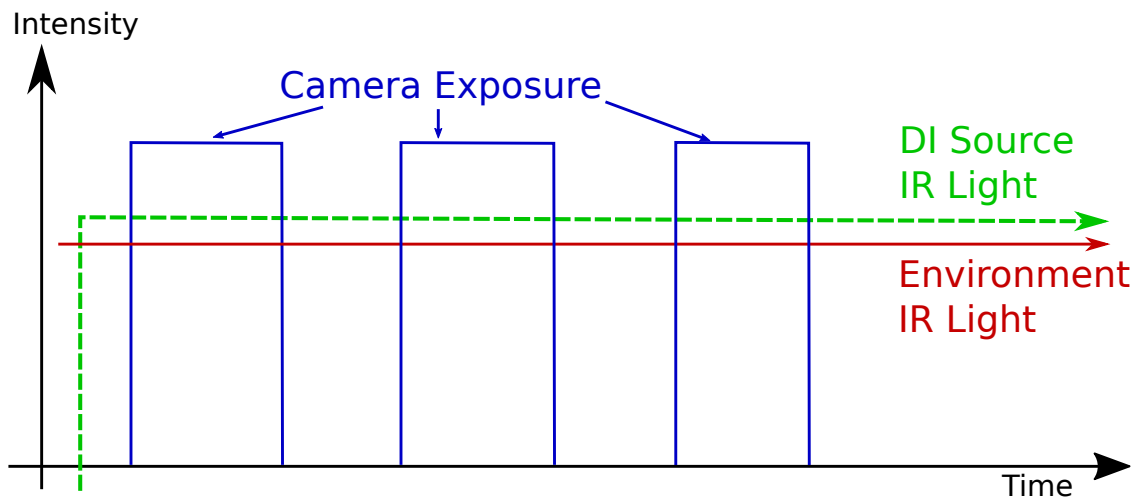


Figure 3.9: Sketch of camera exposure time and infrared source power in DI continuous mode.

mode" in which the infrared sources are driven with much more power than in continuous mode. This pulsed mode, however, requires a cool-down phase before the infrared source can be powered on again. Without that cool-down phase, the infrared source would overheat and quickly get damaged. Optimizing the relationship between infrared energy usage and camera exposure time would also optimize the contrast of the captured images. Therefore, the pulsed illumination approach proposes to overdrive the infrared sources in a pulsed mode in combination with synchronizing the camera exposure duration to fall exactly into the over-driven infrared light pulse as sketched in ► Figure 3.10.

Pulsed Illumination requires fine grained control of light source and camera.

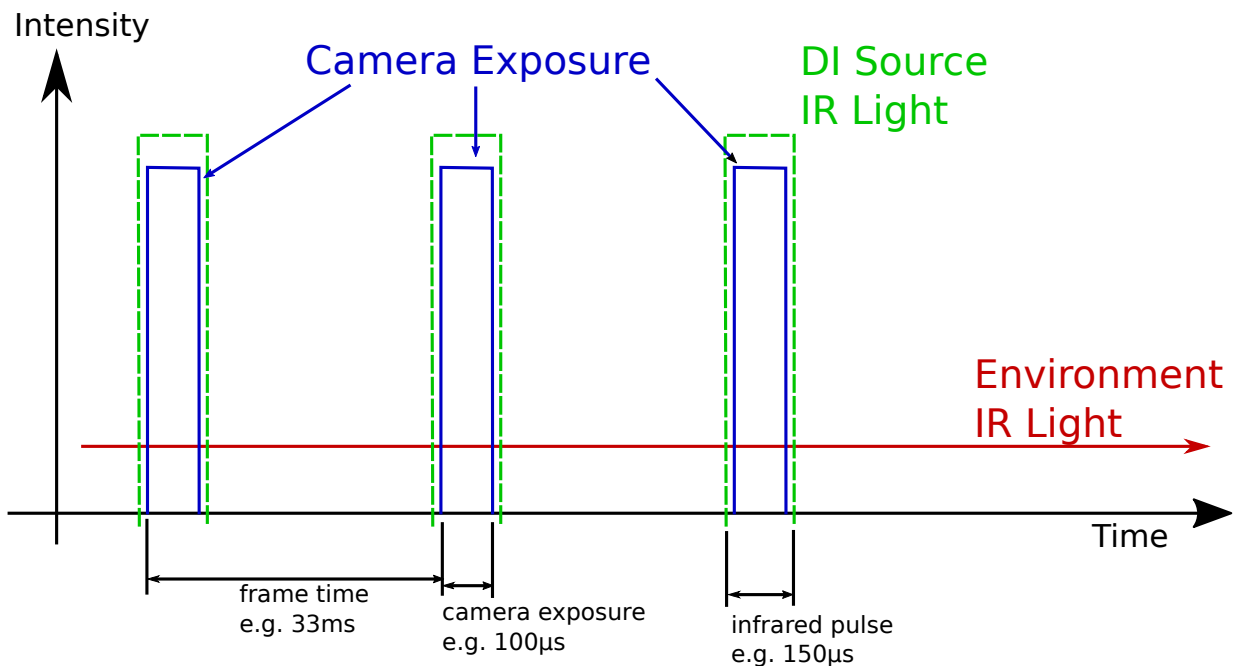


Figure 3.10: Sketch of camera exposure time and infrared source power in DI pulsed mode.

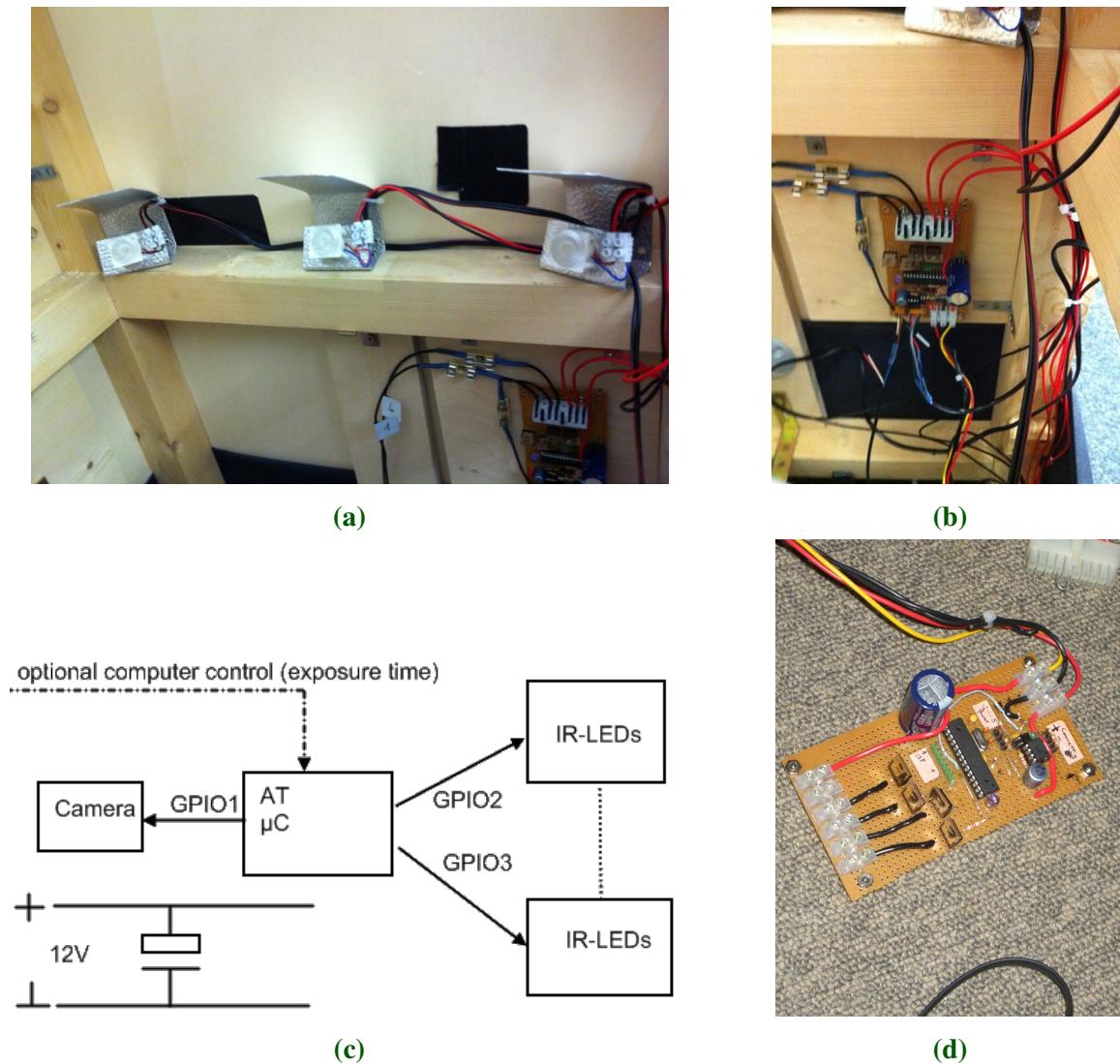


Figure 3.11: (a) High power LEDs used for DI pulsed mode. (c) Sketch of controller board for synchronizing infrared camera and infrared sources. (d) Picture of the controller board (b) mounted in the tabletop interior.

As part of the research for this dissertation, the author applied and evaluated the Pulsed Illumination approach and optimized the components in terms of infrared sources and camera. Of the many infrared sources that we experimented with, the best results were achieved with infrared LEDs of type Vishay TSHA5502 and TSHA6503 (continuous mode 100mA, pulsed mode 1.5 - 2.5A) and integrated high power LEDs of type ACULED®VHL™IR ACL01-SC-III-005-C01⁵ (continuous mode 700mA, pulsed mode 1 - 1.5A), cf. ► Figure 3.11a. We used infrared cameras of type PointGrey FireFly MV FireWire 1394a⁶, which provided an easy to access interface to trigger the camera exposure from an external source. In order to synchronize cameras and infrared sources, we developed

Suitable Pulsed Illumination components and details from our experience are given in the Appendix.

⁵ <http://www.excelitas.com/Pages/Product/ACULED.aspx>

⁶ <http://www.ptgrey.com>

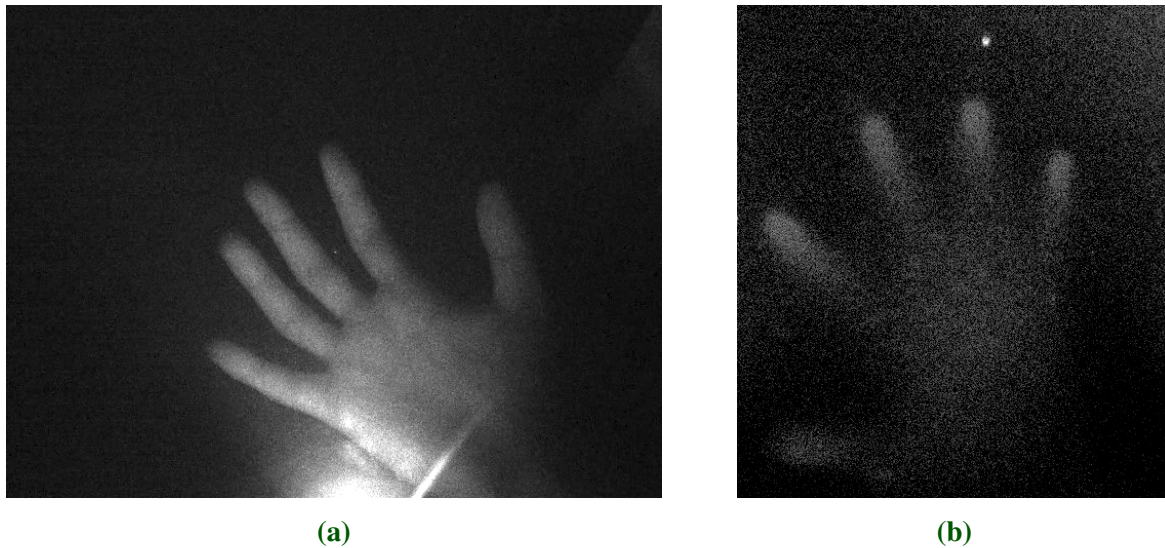


Figure 3.12: (a) A pulsed DI image without diffuser plate. (b) A pulsed DI image with diffuser plate.

a controller board on the basis of the simplified sketch in ► Figure 3.11c. The controller board (► Figure 3.11b, 3.11d) featured an Atmel®ATmega8 micro-controller⁷, which also made sure that power phase and cool-down phase of the infrared sources were in compliance with the according specifications. The full schematic diagram (► Appendix A.2), circuit board layout (► Appendix A.3) and source code (► Appendix A.1) for the controller board is provided in the Appendix for further reproduction.

Our experiences with the Pulsed Illumination approach were rather positive in terms of more stable image contrast when environmental light increases. The pictures in ► Figure 3.12 show examples from our test setup (for daylight conditions) where the background is almost completely black due to the short exposure time. ► Figure 3.12a shows an example without a diffuser plate and ► Figure 3.12b shows an example with a diffuser plate between the camera and the fingers above the sensor surface. However, we also experienced a slightly increased noise level due to the short exposure time, which required an appropriate smooth filter to remove the image noise. Furthermore, we initially had only 4 high power infrared sources mounted in the tabletop for a 30-inch sized tabletop surface. The amount of infrared light turned out to be insufficient and resulted in images that were too dark for the image processing algorithms to provide stable recognition. After upgrading to 6 high power infrared sources, the noise level decreased and the infrared images became much more reliable. Adding more infrared sources merely involved two steps, that is, mounting more sources into the tabletop and attaching them to one of the controller's switches (MOSFET IRF3704). Hence,

Pulsed Illumination is more robust against environmental light changes.

Pulsed Illumination images contain more noise.

Pulsed Illumination is scalable to environmental light.

⁷ <http://www.atmel.com/devices/atmega8.aspx>

this approach offers easy scalability to environments with a higher amount of ambient light by just increasing the number of light sources.

Another approach that is worth mentioning is known as DSI (Diffuse Surface Illumination), which provides a special feature over other the technologies of category DI. In theory, DSI combines the benefits of the category DI with the benefits of the FTIR category's high contrast blob images as can be seen in ► Figure 3.7c. That is, DSI enables finger input property recognition on high contrast blob images as well as recognition beyond the blob images (e.g., marker detection). The idea behind DSI is attributed to a specific acrylic material called EndLighten⁸, which reverts the behavior of FTIR acrylic plates. Instead of catching the infrared light within the acrylic, EndLighten acrylic distracts the infrared light to both large surfaces of the acrylic plate. Thus, the acrylic itself functions as an infrared light relay. From our experience, however, implementation of a working and robust DSI setup is more difficult than the previously discussed approaches. Due to the EndLighten acrylic, much more light energy than in the previous approaches is required to achieve reasonable contrast in the captured images. This, in turn, increases power consumption and heat output of the whole system resulting in the previously discussed issues. A more recent extension of the DI technology further allows recognition of users' fingerprints out of the finger contact areas for identification of users by means of high-resolution cameras and special glass fiber materials as demonstrated in Fiberio [Holz and Baudisch, 2013]. Holz and Baudisch [2013] have shown useful applications of users' fingerprints for tabletop interaction. However, implementation of this extension requires special components and appropriate technical modifications. Thus, this extension is yet not much widespread.

Overall, even though DI systems are more difficult to build, they are as popular as FTIR systems due to the advanced recognition features that go beyond finger input properties, for example, recognition of fiducials or markers. As a general rule of thumb for all DI-based approaches, the more infrared energy employed in the system, the better the resulting infrared images and recognition results.

3.1.2.2 Non-Optical-Based Approaches

Non-optical-based approaches include technologies that do not operate on camera images as sensor data. Those include today's widespread capacitance-based sensor surfaces, resistance-based surfaces, and the DiamondTouch-based tabletop technology. Each of the technologies will be briefly discussed through the lens of supported finger input properties.

⁸ PLEXIGLAS EndLighten (Evonik 0N001 - 0N003)

Idea of DSI is to combine FTIR and DI.

Basis for DSI is a specific acrylic material.

DSI requires much more infrared light than the other approaches.

DI extension to recognize fingerprints.

DI offers the most potential for tabletop interaction but requires sufficient amount of infrared light.

Non-optical-based approaches do not require cameras.

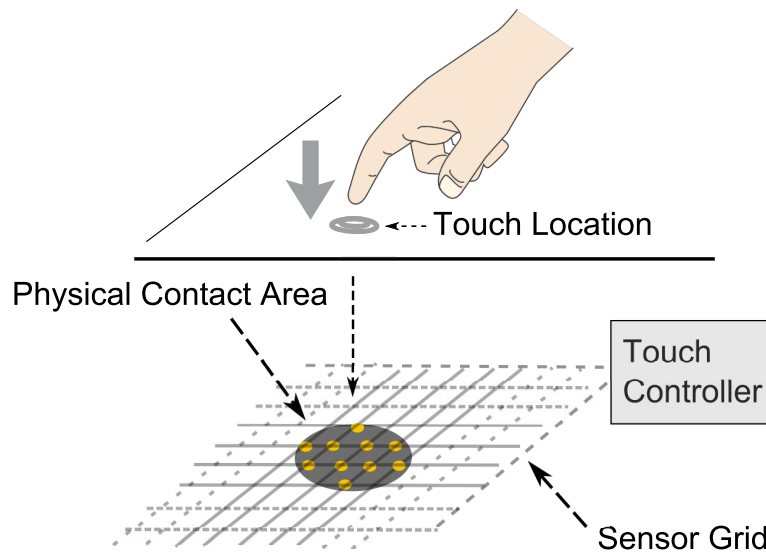


Figure 3.13: Sketch of the contact area on the sensor grid of a capacitance-based sensing surface.

What makes the category Capacitance unique?

CAPACITANCE. Capacitance-based sensor surfaces detect finger contact areas through observing the changes in many capacitance measures that are evenly distributed across the whole sensor surface. For this purpose, the sensing surface features an embedded quite dense and close-meshed lattice-like layer, which is made of conductive material. This layer allows measurement of capacitances in combination with a touch controller as sketched in ► Figure 3.13. If a conductive material, such as the human skin, comes close to or has contact with the surface, then the capacities in the embedded lattice at the contact locations change. Since the embedded lattice is arranged quite dense, not only one measured point changes but all measured points that the contact area covers change. All measured points together can then be approximated to the physical contact area as discussed for ► Figure 3.2b. An important character of capacitance-based sensing surfaces is their ability to detect multiple physical contact areas at the same time due to the composition of the embedded lattice-like layer in combination with the touch controller.

Capacitance-based sensing enables multiple touch contacts on the surface.

Recognition of finger input properties.

The capacitance-based approach basically enables recognition of finger input properties that are also available for the category FTIR because the approximated physical contact areas are comparable to the blob images of the category FTIR. However, the actual set of finger input properties available to applications is limited by the employed touch controller and system software. For example, there are sensor surfaces that provide the finger orientation or the physical contact area by means of the major and minor axis in combination with a value that indicates the physical contact size (e.g., Google Nexus 10⁹). There are also sensor surfaces that drop those finger input properties even if the touch controller enables recognition of such properties. In principle, the measured capacitance points en-

Limitations of Capacitance.

Capacitance measures potentially provide more finger input properties.

⁹ https://de.wikipedia.org/wiki/Nexus_10

able software applications to recognize the shape of a physical contact area (e.g., ellipse or rectangle shaped). However, most commercially available sensor surfaces and touch controller consider only finger touch contacts and always assume ellipses as the contact shape. Though, Xiao et al. [2015] demonstrated that the access to the capacitance measures of a touchscreen enables recognition of more finger input properties for interaction than most touch controllers provide.

DT. The category DT offers a unique feature that is realized solely by the MERL¹⁰ DiamondTouch tabletop [Dietz and Leigh, 2001]. This tabletop technology has often been used for research studies due to its early availability as one of the first interactive multi-touch tabletops. Meanwhile, this technology is commercially distributed by the company Circle Twelve Inc.¹¹ DiamondTouch's sensing principle is also capacitance-based but much different from the former discussed capacitance-based sensor surfaces. The sensing surface has a transmitter antenna array embedded within the thin surface, which is structured as a grid with multiple rows and columns that allow a sensing resolution of 0.4mm [Forlines et al., 2007]. Each transmitter antenna (controlled by a transmitter unit) emits its own signal that can be sensed by receivers, which are fabricated into mats on which users have to sit or stand on [Dietz and Leigh, 2001].

The DiamondTouch supports max. 4 receiver mats, which are usually placed on the seats around the tabletop. In this configuration, the user plays the role of a proxy through which the antenna array capacitively couples with the receivers, hence it requires users to keep in contact with the receiver mats during the whole interaction. Since each user sits on a different receiver mat, the DiamondTouch tabletop assigns touch contacts based on receiving mats to users and provides the finger contact property "user identifier". This property cannot be recognized by other technologies without additional extensions or instrumentation such as, for example, the "Carpus" approach by Ramakers et al. [2012]. The Carpus extension provides the user identifier in a non-intrusive way for every technology in ► Table 3.1. As a short clarification remark, the sensing approach of the Depth category is able to distinguish touch contacts of different users but cannot reliably identify a touch contact's user in the first place, for example, for two or more users standing next to each other at the same side of a tabletop.

A peculiarity of this technical approach occurs when two users are in contact with each other while one of them is touching the sensing surface. In this case, the antenna signals are capacitively coupled to two receiver mats at the same time, resulting in two touch events at the same location by two different users, though only one has physically touched the surface [Fikkert et al., 2009].

Category DT contains only the MERL DiamondTouch tabletop.

What makes the category DT unique?

Category DT enables user identification.

Category Depth enables user distinction.

Limitations of the category DT.

¹⁰Mitsubishi Electric Research Laboratories

¹¹<http://circletwelve.com>

Compared to today’s capacitance-based sensing surfaces, DiamondTouch’s sensing resolution is coarse-meshed and not sufficient to support other contact area related finger input properties. Even though the DiamondTouch supports multi-touch, researchers [Collberg et al., 2003, Haller et al., 2010] have reported that more than one touch contact of the same user at the same time can result in ambiguous touch locations due to the row/column antenna design.

RESISTANCE. Resistance-based sensing surfaces were the preferred technology for mobile PDAs¹² or laptops in the past. Nowadays, capacitance-based technologies have superseded resistance-based technologies in those domains and displaced resistance-based touch technologies to niche domains in which their strengths, such as pressure sensitivity, are required.

The basic composition of resistance-based sensing technologies has two translucent and electrically conductive layers embedded into the sensing surface whereof the outer one is flexible and the inner one is rigid. Both layers are separated by means of multiple tiny insulating dots [EloTouch, 2015]. Human fingers create touch events by exerting pressure on the sensing surface, which in turn changes the electrical resistance between the conductive layers. A touch controller evaluates this change and calculates the corresponding x/y-position of the touch contact.

Compared to all other sensing categories in ► Table 3.1, a sole finger touch without pressure on the surface is not sufficient for touch input. This fact, however, can be beneficial because it enables interaction with arbitrary objects and materials such as a stylus or gloves. In addition to that, the touch controller also provides the amount of pressure that was applied to the surface by evaluating the value of the change in resistance.

The functional principle of resistive touch technologies inherently allows a maximum of two touch contacts at the same time. Hence, this technology is mainly employed for application domains where pressure activation is required or single touch is sufficient or even required, such as public terminals at airports or train stations.

Overall, the amount of available finger input properties is limited to those two that were discussed before. Furthermore, large tabletop sized sensing surfaces are hardly available and much more expensive than sensing surfaces of the other technologies.

¹²Personal Digital Assistant

Basic composition of resistance-based sensing.
What makes the category Resistance unique?
Recognition of finger input properties.
Limitations of Resistance.

3.2 Usage and Potential of Input Properties

So far, we have established an overview of the most widespread enabling tabletop technologies together with their technical features to provide finger input properties. In this section, we elaborate on the usage and potential of the identified finger input properties in ► Table 3.1 for tabletop interaction and applications. If we think of platform-independence, an inevitable question that arises is which of the properties should be used and how they should be used to realize interaction techniques and user interfaces. If we think of platform-dependence, developers and designers should be able to assess the finger input properties in view of availability, reliability, and precision for different tabletop sensing technologies.

As we have already ascertained, the x/y-position of a touch contact is provided as the most reliable property by all sensing technologies and therefore prominently used by tabletop applications. Many of the applications even exclusively use the x/y-position for interaction. Tabletop applications will (and should) continue using this finger input property prominently due to the strong and natural affordance to touch visual objects on the tabletop surface with fingers. Moreover, many people are used to this interaction by reason of previous or daily experiences with user interfaces or touchscreen devices.

LIMITATION. The category "Resistance" and "DT" are omitted in the following discussion. These technologies offer only few finger input properties due to the former discussed technological background. Tabletop applications are currently much more likely to encounter systems based on the other discussed technologies. When considering the remaining finger input properties in ► Table 3.1, one can notice that the availability of some properties is limited to one category of the technologies (i.e., fingerprint or hand posture). Due to this limited availability, the widespread adoption of those properties is at least questionable. Hence, we focus the discussion on those finger input properties as well as technologies that are widely available for interaction design.

3.2.1 Idiosyncrasies of Touch Interaction

Before we discuss individual finger input properties, developers and designers of touch interfaces should be aware of two idiosyncrasies of touch interaction, which are referred to as "the fat finger problem" [Siek et al., 2005] and "the exit error problem" [Tuddenham et al., 2010] in existing research literature.

The fat finger problem paraphrases two issues that originate from the large size of a human finger contact area in comparison to the much smaller size of an

Which of the finger input properties should be used and how?

How is the availability, reliability, and precision of finger input properties on different technologies?

X/y-coordinate provided by all technologies.

Category Resistance and DT are limited, thus omitted in further discussions.

The fat finger problem.

x/y-location on a sensing surface. This relationship introduces occlusion of the sensing surface caused by the human finger as well as imprecision in pixel-wise selection through the sensing surface caused by the combination of the large contact area of a human finger together with finger occlusion. Tabletop applications primarily solve this problem by designing the size of selectable objects to be large enough for touch interaction. Large objects solve finger occlusion for selection tasks and eliminate the need for selection at pixel precision. For selection at pixel precision, researchers have presented several approaches based on the shift or scale of the occluded areas as extensively discussed by Müller-Tomfelde [2010].

The exit error problem.

The exit error problem is caused by the short moments in which human fingers release the contact with a sensor surface. In such moments, the finger contact area shrinks until its size falls below a certain threshold. However, the shrinking contact areas result in x/y-positions that often varies (at least) a little [Tuddenham et al., 2010]. This behavior is not a problem for user interface elements that cannot be moved, such as buttons or checkboxes, as those are mostly triggered by touch-down/up events. For pixel-wise drawing tasks or user interface elements that are movable (e.g., picture objects) or adjustable (e.g., sliders or scrollbars), the small variations of the x/y-position lead to inadvertent changes of values or selections. Tabletop applications need to consider this behavior and provide means to fix a selection, for example, by including catching or latching marks.

3.2.2 Size / Shape of Contact Area

Availability of contact area's size and shape.

The finger input properties "size" and "shape" of contact areas are quite simple to determine and are related to each other. Due to this fact, they are provided by most of the tracking applications for digital tabletops. On mobile capacitance-based touchscreens, the touchscreen controllers provide sensor data that enable calculation of those properties [Xiao et al., 2015] and many of such platforms provide those to applications (e.g., Google Nexus 10).

Contact area shapes are provided as ellipses with major- and minor-axis.

SHAPE. The shape of a physical finger contact area roughly consists of an elliptic or circular shaped and dense aggregation of pixels or measuring points around a determinable mass center, cf. ► Figure 3.7a. For this reason, every tracking system assumes the shape of a contact area to be an ellipse and approximate the shape using the ellipse parameters, that is, major-axis and minor-axis, with the center position of the contact area as the reference, cf. ► Figure 3.2b. Due to the relation between shape and size of a contact area, they are often used together.



Figure 3.14: SurfacePoker: A poker game that makes use of hand shape gestures.

SIZE. The size of a finger contact area has two technological dependencies in terms of hardware and software. First, depending on the underlying sensor technology, the size of a contact area is determined by either the number of pixels or the amount of measure points within the whole finger contact area. Consequently, the resolution (i.e., density of pixels or measure points) at which the sensing technology operates has a strong influence on the range and achievable precision of the contact size value. Second, the interpretation of the contact size value depends on the tracking software that is employed. Some tracking systems perform a normalization of the value or a transformation into a physical unit. For instance, the Microsoft PixelSense [Microsoft, 2011] calculates a value that matches the physical unit "square inch" while the TUIO specification [Kaltenbrunner, 2009] recommends a normalization into the range between 0.0 and 1.0. In addition to the technological dependencies, the finger contact area's size and its variance are subject to the size of user's hand and the particular finger of a hand. For example, fingers of small children lead to smaller finger contact areas than the fingers of adults and the variance of the thumb's contact area is larger than the variance of the little finger's contact area.

These circumstances can be taken into account by using only a few thresholds and by dynamically adapting the thresholds. For example, the SimPress technique [Benko et al., 2006] makes use of the finger contact area's size to realize a click mode. The first touch contact selects an object while a subsequent press-action on the object, which is detected through a large increase of the finger contact area size, leads to a virtual lock on the selected object. As an unwanted side effect of this technique, the center position of a finger contact area moves during performing the press-action, which needs to be compensated as proposed in the SimPress technique. Another example is called ShapeTouch by Cao et al.

Contact area size depends on sensing resolution.

Interpretation of contact area size values depends on tracking software.

Contact area sizes are different for different users, e.g., children vs. adults, and different fingers of a hand.

Using contact area sizes to distinguish between touch and click.

Using contact area sizes to extend gestures.

[2008], which shows how the shape and size of contact areas can be employed to directly influence gestures. In the tabletop poker game SurfacePoker [Dang and André, 2010], we included the size and shape of the contact areas that the hands of a user create when forming a shielding gesture on the tabletop surface, cf. ► Figure 3.14. This gesture established a physical barrier that prevented opponents from seeing one's game cards' if the tabletop game uncovers the cards. The usage of finger contact size has also been proposed for mobile touchscreen applications and may be realized for tabletop applications in the same form. Boring et al. [2012] or Bonnet et al. [2013] makes use of the contact area size in combination with the center position to distinguish between the rolling of the thumb and a swipe gesture with the thumb.

3.2.3 Pressure / Pseudo-Pressure / Force

Only resistance-based technology can sense pressure without further extensions.

Pressure-alike values can be derived from contact sizes.

Limitations of derived (pseudo-) pressure values.

The term "pressure" actually denotes the physical pressure conducted by a human finger on a sensing surface. Among the sensing technologies in ► Table 3.1, only the resistance-based technologies can actually sense the physical pressure. The other technology categories cannot sense physical pressure without being extended with dedicated pressure sensitive constructions, materials, or liquids [Hilliges et al., 2008]. However, a pressure-alike value can be derived by leveraging the finger contact area's size since the contact area on the sensing surface spreads around the center position as the finger touches harder on the sensor surface. Compared to a pressure value resulting from physical pressure, the increase of the derived pressure value is limited by a threshold at which the increase stops or changes only marginally no matter how much more pressure is actually applied by the finger, whereas the physical pressure still can be increased over that derived pressure threshold. Hence, the term "pseudo-pressure" as proposed in [Arif and Stuerzlinger, 2013] is more appropriate for such derived pressure values with respect to the differences to directly measured pressure values. Buxton [2007] also mentioned this difference and denoted such derived pressure values as "degree of contact".

How many pressure levels should be used?

Six pressure levels seem optimal for stylus interaction.

Equally important to the technical ability to sense pressure is the human ability to exert pressure and distinguish between different pressure levels. Early research by Ramos et al. [2004] investigated human skills to perform different levels of pressure on a sensing surface. They found that dividing the pressure space into six levels was optimal, but their results are limited to a certain technology as they conducted the experiment with a stylus on a resistance-based sensor surface, which is different from exerting pressure with a human finger. Distinguishing between different pressure levels requires users to detect or estimate (through tactile feedback) how much pressure they actually exert. In case of a human finger, users primarily make use of their mechanoreceptors of the

finger contact areas with muscles in finger and hand [Bolanowski et al., 1988, Gescheider et al., 2002]. In case of a stylus, the finger has no contact with the sensor surface, but the exerted pressure is conducted through the tip of the stylus. Recent research by Arif and Stuerzlinger [2013] that focuses on pseudo-pressure with human fingers could distinguish reliably at least between two pressure levels due to the limitation of their recognition approach. Commercially available pressure-sensitive touchpads and touchscreens, such as Apple's ForceTouch or 3D-Touch, mostly make use of two different pressure levels even if much more levels can be technically distinguished. In terms of large tabletop sized sensor surfaces, commercial products that support pressure or pseudo-pressure, for example, the FlatFrog¹³ displays, promote up to 1000 different pressure levels that can be distinguished.

At least two pseudo-pressure levels can be reliably recognized.

The HCI research literature offers significant and promising examples for tabletop applications using pressure or pseudo-pressure for interaction techniques that still needs to be adopted by developers and designers. To name a few examples, Kim et al. [2010] made use of the pseudo-pressure to enhance security for PIN¹⁴ authentication. Keller et al. [2012] added virtual weights to graphical objects that had to be operated based on different pressures. An interesting text-entry extension demonstrated by Brewster and Hughes [2009] mapped soft press on a virtual keyboard to lowercase letters and hard press to uppercase letters. Arif and Stuerzlinger [2013] showed how to add modes to touchscreen text-entry mechanisms based on the pseudo-pressure and Boring et al. [2012] employed pseudo-pressure of the thumb to switch between different modes within a map application. Interaction techniques using pressure or pseudo-pressure should be carefully designed with user expectations in mind since users are not yet used to have different effects by performing touch with different pressures. Exploiting more than two pressure levels may confuse users, as touch interaction using pressure is still not widespread, which, however, may change in the future.

Pressure and pseudo-pressure enable promising interaction techniques.

TAPPING FORCE. An input property that is similar to pressure or pseudo-pressure is the tapping force, which is defined as the force that a human finger applies when the finger lands on the sensor surface. It can be determined by means of pressure sensing. In comparison to pressure, tapping force considers only the landing moment on the surface instead of the pressure applied during the whole contact duration of a finger with the surface. Pedersen and Hornbæk [2014] made use of 4 microphones as sensors to determine the tapping force and found in a study that users are able to reliably (99%) perform at least two force levels. Similar to the usage of pressure, interaction techniques employing the tapping force should be designed with care as users are usually not familiar with different behavior caused by different tapping forces.

¹³Pressure Sensitive FTIR: <http://www.flatfrog.com>

¹⁴Personal Identification Number

3.2.4 Finger posture

Technologies capable
of sensing finger pose.

Interaction techniques
using finger pose.

Adoption of finger
pose.

A finger pose is given by pitch, yaw, and roll angle (x,y,z-tilt) of the finger during contact with a sensor surface. Only two sensor technologies can recognize finger pose without further instrumentation. The category Depth can detect the finger pose with only a single frame of sensor data as depth cameras provide depth images that contain the whole hand including the fingers. For the other category, that is, capacitance-based sensor surfaces, Xiao et al. [2015] recently demonstrated how to estimate pitch and yaw of a finger pose by means of the capacitance measurements from off-the-shelf consumer touchscreen devices (i.e., Samsung Galaxy S4 smartphone, LG 'G' smartwatch). Existing research works have proposed several promising interaction techniques that include or would benefit from the inclusion of the finger pose. To name a few examples, Holz and Baudisch [2010] found that knowing the finger pose (which is considered for their generalized perceived input point model) can significantly improve touch input accuracy. Roudaut et al. [2009] employed finger roll in their "MicroRolls" technique to successfully extend thumb gestures for copy and paste tasks. Many examples and demo applications were presented by Xiao et al. [2015], which made use of the finger pose to control pan, zoom, or rotate operations as well as 3D manipulations. Finger pose is a finger input property that has not been widely adopted yet for interaction techniques neither for tabletop interaction nor for touch interaction with mobile touchscreens. Hence, many users would not realize the existence of this interaction channel in the first place. Therefore, integration of finger pose for tabletop interaction should not be overused and only applied where appropriate.

3.2.5 Remaining Input Properties

The remaining finger input properties, that is, the finger orientation and hand distinction, have not received much attention in the existing research literature, for example, as indicated by Wang and Ren [2009] in terms of the finger orientation. At the time when our research on finger input properties was conducted, there were a few works that proposed technologies based on additional cameras (e.g., Dohse et al. [2008], Do-Lenh et al. [2009]), which basically were able to recognize hand distinction or finger orientation. However, no works have further investigated these input properties for interaction techniques even though both properties provide rich possibilities to realize interaction techniques. This thesis, therefore, analyzes and discusses both properties in detail and present research contributions that address finger orientation and hand distinction for tabletop interaction. In the following, both finger input properties are dedicated its own sections with updated revisions of the related paper publications by the author.

3.3 Finger Orientation / Direction

Contribution Statement: An earlier version of this section's content has been published as a peer-reviewed full paper [Dang and André, 2011] at the 13th IFIP TCI3 conference on Human-Computer Interaction INTERACT 2011. This section presents an algorithm and evaluation for precise calculation of finger orientation for camera-based tabletops that employ the Diffuse Illumination sensing technique. At that time, approaches with demand on high precision were known only for FTIR-based tabletops.

https://doi.org/10.1007/978-3-642-23765-2_28

The finger orientation represents the direction in which a finger points while interacting with applications on a tabletop surface. ► Figure 3.15 (or ► Figure 3.2b) exemplarily sketches the finger orientation of a finger contact area. Usually, the finger orientation is given as a degree value within the range $0^\circ \dots 360^\circ$. Thus far, finger orientation has rarely been used in interaction techniques for multi-touch tabletop applications as also claimed by Wang and Ren [2009]. Obviously, the usage of finger orientation opens up rich opportunities to ease or enrich tabletop interaction due to human's inherent understanding of directions and orientations. Humans start to develop an understanding of directions already at the early age of a child. Humans then internalize this understanding by repeated recall and use every day and for many tasks in the real world. The usage of this understanding becomes so natural that we automatically make use of directions to complement or convey our intention, for instance, when pointing at objects or indicating a direction.

Within human-human interaction, the finger orientation is even meaningful if used as the sole communication channel, but most of the times the finger orientation accompanies other communication channels or interaction modalities. For example, pointing at an object with the index finger is often sufficient to indicate a distant object. Usually, using speech at the same time serves to additionally describe the object in more detail. A more concrete example would be: pointing at a bottle of water in tandem with pronouncing the words "the bottle of water over there". Often, the pointing gesture happens automatically without having to think about. A similar behavior can be observed for people who tell foreigners the way to a destination. During the whole conversation, people often automatically show directions through hand or finger gestures so as to complement and clarify the spoken instructions. If both (speakers) speak a different language,

Definition of finger orientation.

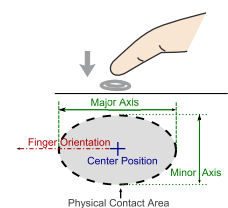


Figure 3.15: Finger orientation in relation to the center position of a finger contact area.

Directed gestures through hand or finger orientations are inherent parts of human-human interaction.

then oriented gestures together with symbolic or iconic gestures represent the main information channel. Apparently, employing finger orientation to extend or complement interaction techniques would offer valuable potential for a more natural form of tabletop interaction because finger orientations are an inherently integrated aspect of human's interaction in and with the real world.

Certainly, one reason why finger orientation has not yet often been considered for tabletop interaction may be the scarce support of finger orientation in freely available tracker software, such as [NUIGroup, 2014a] or [NUIGroup, 2014b]. Reliable methods to determine finger orientation unambiguously and with high precision would foster integration of finger orientation recognition into tracker software. Approaches to determine finger orientation are known for sensing technologies of the category FTIR [Wang et al., 2009] as well the category Depth [Malik and Laszlo, 2004]. However, approaches for the sensing technology category DI have been neglected. Therefore, we present and discuss an algorithm and evaluation for precise calculation of finger orientation for camera-based tabletops that employ the Diffuse Illumination sensing technique. The evaluation provides proof for the precision of the approach by means of four experiments, which show a recognition rate of 97.8% at a tolerance of 15° . At a smaller tolerance of 5° , our approach still provides an acceptable recognition rate of 75.24%.

This section is structured as follows: In the next two subsections, we motivate the use of finger orientations for natural interaction by giving design examples of user interfaces and interaction techniques. Following up, ► Subsection 3.3.3 introduces the technical background of our approach followed by a description of a naive approach and our new approach for the detection of finger orientation. In ► Subsection 3.3.5, we present recognition rates for the presented approaches and discuss benefits and issues.

3.3.1 Usage of Finger Orientation

Incorporating finger orientation into the design of natural interaction offers many chances for improvements related to several issues that pertain to touch user interfaces and interactions, that is, manipulation, occlusion, selection, and adaptation. The design of both, that is, interaction techniques and user interfaces, can profit from including finger orientation as exemplified in the following.

MANIPULATION. Wang et al. [2009] presented a variety of interaction techniques, such as orientation-sensitive widgets, that would be enabled by robust techniques to recognize finger orientation. Such widgets take the finger orientation into account and enable selection of different functions combined with

Scarce support for finger orientation in tabletop tracker software.

No known approach for recognition of finger orientation and DI tabletops.

Benefits of finger orientation for manipulation, occlusion, selection, and adaptation.

Orientation-aware widgets.

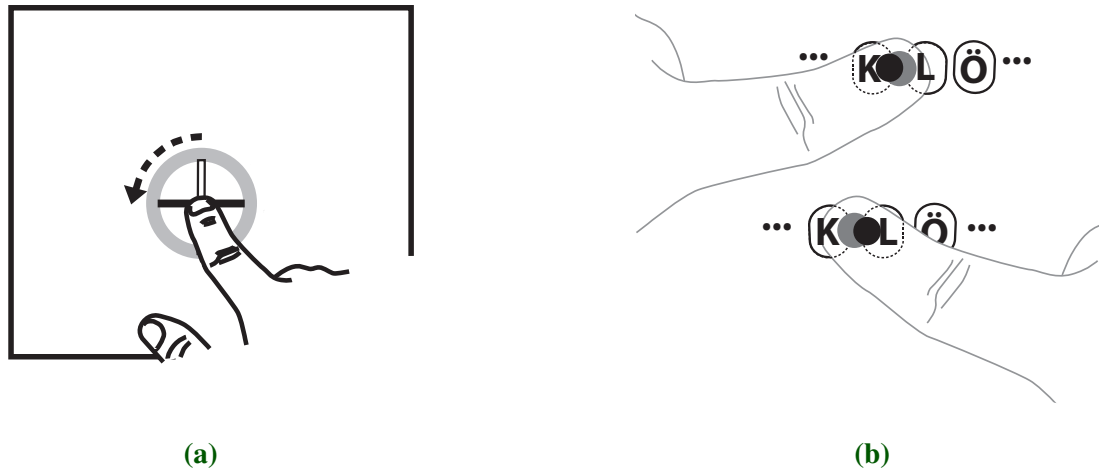


Figure 3.16: Finger orientation usage examples: (a) A rotary switch with several latches; (b) Adaptation of reported input point for virtual keyboards.

adjustment of parameters, which require only less space for user interaction. Another example of manipulation techniques – that include finger orientation – becomes apparent on a rotary switch that offers only two states for pinning or releasing an object in the workspace, cf. ► Figure 3.16a. Users have to spin the switch with a finger in order to lock or release the virtual object, thus avoid unintentional triggering of the switch's state, for example, if users inadvertently place a finger on the switch.

Rotary switch.

OCCCLUSION. Information about finger orientations would allow applications to determine areas that are potentially occluded by the hand and adapt the display of graphical objects accordingly. For instance, occluded user interface widgets could be re-oriented or displaced in such a way that they are visible to the user as best as possible. Furthermore, techniques such as "occlusion-aware pop-ups" or "occlusion-aware dragging" presented for pen or stylus interaction in [Vogel and Balakrishnan, 2010] could be realized for interaction with bare fingers by means of the finger orientation. Thereby, hierarchical menus or tooltips might appear in the non-occluded area and occluded text segments could be replicated to callouts in non-occluded areas to support selection tasks.

Presenting content on non-occluded areas.

SELECTION. By inclusion of the orientation of fingers, users might be able to use their fingers of either one hand or both hands for pointing at objects displayed on the surface, allowing them to select distant objects in a more natural manner, cf. ► Figure 3.17a. Users could also use the thumb and the index finger of one hand together to span an open angle for object selection, cf. ► Figure 3.17b. A further option opens up if users make use of both hands to span two open angles as sketched in ► Figure 3.17b. Both selected areas may intersect with each other and create an intersection area, which narrows down a certain area for object selection.

Spanning of areas for distant selection.

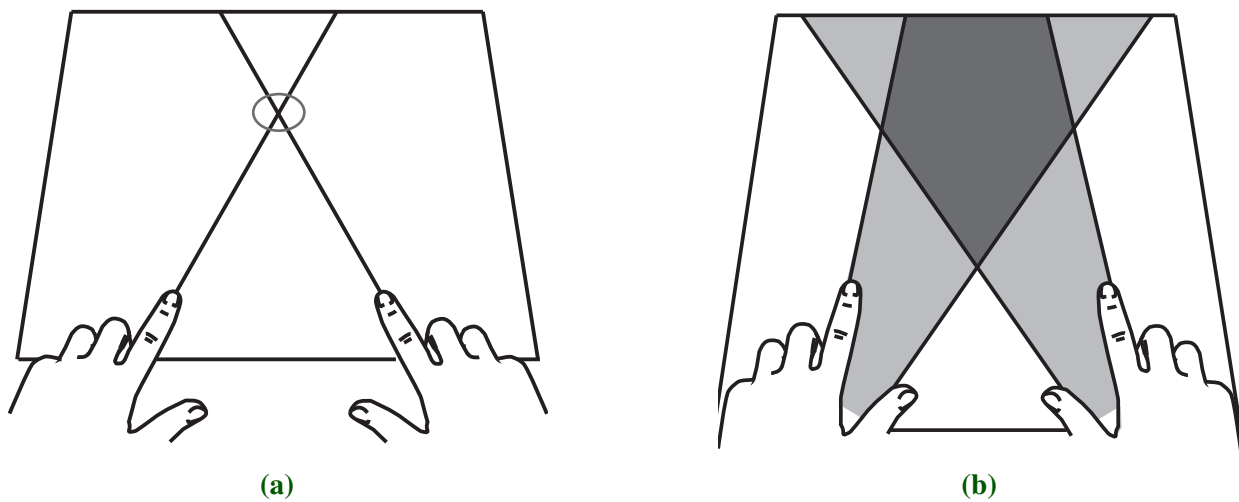


Figure 3.17: Finger orientation usage examples: (a) Distant object selection using cross-hair through oriented lines; (b) Fingers spanning open angles for area selection.

ADAPTATION. Vogel and Baudisch [2007] showed that the reported input point of a finger touch contact differs from the user's intended target location by an offset. This offset is due to the fact that the users' perceived input point is different from the real target location. Based on this observation, they suggest an adaptation approach to correct the reported input point by such offsets in order to increase selection precision. Here, the finger orientation would be necessary for providing the direction in which the adaptation should be applied.

Improve selection accuracy by means of offset adaptation.

As an example, the interaction with a virtual keyboard on small touchscreens would benefit from such an adaptation as sketched in ► Figure 3.16b. Without adaptation, the reported input point (gray colored circle) would result in ambiguous key selection, though the reported input point would better fit the perceived input point with an offset adaptation (black colored circle).

From a pragmatic point of view, user interfaces and interaction techniques that integrate finger orientation would help mitigate one of the main issues in natural interaction, namely the arm fatigue issue [Wigdor et al., 2007, Yee, 2009]. By reducing the overall amount of hand and arm movements for manipulation or selection tasks, users' hand and arm fatigue would be diminished as well. Not only single interaction techniques can be improved, but also higher level recognition tasks, such as the distinction between one- and two-handed interactions [Dang et al., 2009], would be enabled by reliable detection of finger orientation. In [Dang et al., 2009], we presented a mechanism to distinguish hands based on only the location and orientation of finger contact areas.

Reduce arm fatigue through interaction techniques with fewer movements.

3.3.2 Related Work

Many research works have been conducted in the past addressing finger orientation recognition in 3D space employing multiple cameras or color images, for example, approaches proposed in [Jennings, 1999, Hung et al., 1998, Tianding, 2008], whereas only a few works covered detection of finger orientation for 2D touch sensor technologies or technologies that are based on infrared images. We will briefly discuss them next.

Recognition approaches addressing finger orientation in 3D are well known but not for 2D touch technology.

Malik and Laszlo [2004] presented the Visual Touchpad that utilized two color cameras mounted above the touchpad to detect user's hands and fingers. They identified finger positions through computer-vision methods in order to find the fingertips on a hand contour. The hand contour is also used to determine the finger orientation for each fingertip. Their approach is based on color images and a non-occluded and direct view on the hands, which is quite different from prevalent multi-touch sensing technologies employing infrared images and a bottom-up view on the sensing surface. Furthermore, recognition approaches for tabletop technologies with a bottom-up view have to cope with blurred images caused by diffuser plates, which make much higher demands on computer-vision.

Recognition approach using color images and a top-down view on the scene.

Wang et al. [2009] proposed an approach to unambiguously determine finger orientation that is based on the contact areas produced by finger touches. They fit an ellipse into the shapes of the contact areas and use the longer ellipse axis for determination of finger orientations. In addition, they observe the center point variations of the contact areas when a finger lands on the surface to disambiguate the finger orientation. Their approach is suitable for sensing technologies that provide only finger contact areas, for example, sensing technologies of the category FTIR or Capacitance, whereas sensing technologies such as Diffuse Illumination offer more potential for the detection of finger orientation with a higher precision. Apart from the recognition approach, they also show that finger orientation is a useful input property for enhancing user interactions.

Recognition approach using sequences of blob images.

Observing center point variations to disambiguate finger orientations.

A rather simple and inexpensive way to integrate finger orientation in multi-touch tabletop interaction was conducted by Marquardt et al. [2010]. They employed the Microsoft Surface tabletop [Microsoft, 2011] and a glove, which was tagged with several small graphical markers. The tabletop system was able to detect the markers together with the markers' orientation. Thus, the approach allowed the system to derive finger orientations and to identify individual parts of the hand and their orientations. While wearing gloves is contrary to the idea of natural interaction, Marquardt et al. proposed their approach for rapid prototyping of interaction techniques. They also indicated that integrating more properties of fingers and hands provides rich opportunities for interaction design.

Recognition approach using small markers glued onto gloves.

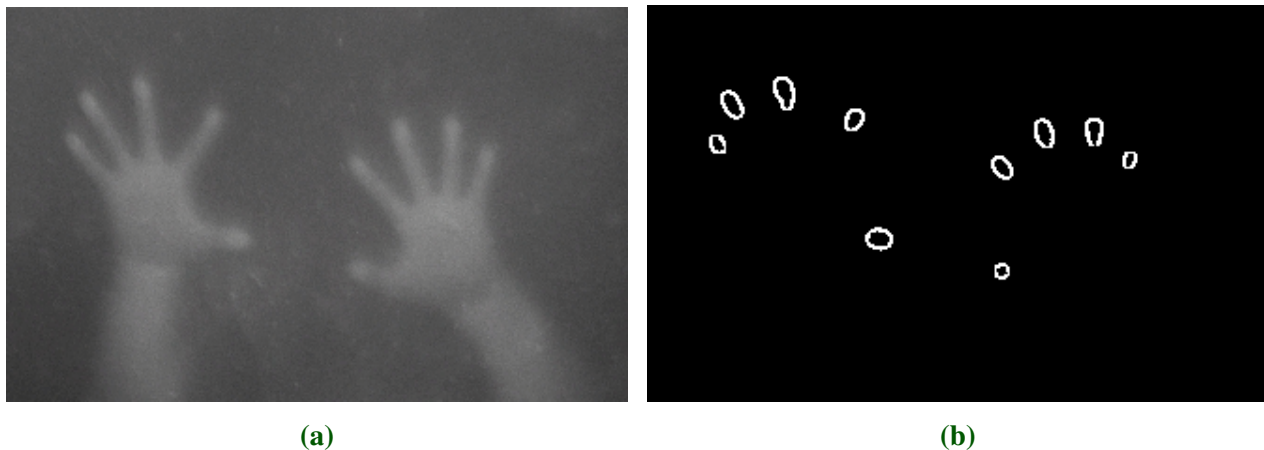


Figure 3.18: (a) Diffuse Illumination infrared image and (b) the corresponding contour image.

So far, we pointed out that finger orientation offers meaningful chances to extend or complement natural interaction. In order to be used for user interaction, finger orientation must be determined reliably and with high precision. For this purpose, we showed existing recognition approaches and discussed their appropriateness for tabletop systems of the category DI. As none of the approaches exploits the benefits of DI tabletops, the next sections present and discuss a simple approach that reliably detects finger orientation for tabletop setups of the category DI. This approach can even detect finger orientations for difficult cases, for example, when fingers touch the surface only slightly.

3.3.3 Diffuse Illumination

The approach presented in this section refers to infrared images from the sensor technologies of the category DI, for example, DI, DSI, or MLLP, cf. ► Subsection 3.1.2.1. ► Figure 3.18a shows an example of such images. Such images offer rich possibilities for object detection through computer-vision methods that take the pixel values and the whole range of the values into account. In such images, the values of contact area pixels (from a finger that touches the surface) represent much more infrared light than pixel values of non-contact areas. Hence, typical computer-vision steps exploit this fact in order to find finger contact areas and to locate the finger coordinates. Usually, a brightness threshold serves to distinguish contact areas from non-contact areas. The steps are roughly comprised of converting raw camera images into blob images based on the brightness threshold and afterwards converting the blob images into contour images. Within the resulting contour images, each contour represents a contact area of one finger that touched the sensing surface as shown in ► Figure 3.18b.

Idiosyncrasies of
images of the
category DI.

Computer-vision steps
to find contours of
contact areas.

Finally, the coordinate of each finger contact can be determined as the corresponding contour's center location.

3.3.4 Determination of Finger Orientation

In this section, we will first outline a straightforward way to determine finger orientation, which serves as the baseline for the performance comparison in ► Subsection 3.3.6. The remainder of this section illustrates our new approach in detail in order to ease integration into tracker applications.

3.3.4.1 A Naive Approach

A naive approach to determine finger orientation relies on the contour image that we discussed before. All the steps of this approach can be easily accomplished with high-level functions of the computer-vision package OpenCV [2015]. Henceforth, we use the term "ellipse-method" to denote the naive approach described in the following.

When considering the contour image in ► Figure 3.18b, we can easily identify each finger contour as an ellipse, representing the finger contact area. Apparently, this fact enables us to fit an ellipse into each detected contour and take the angles between the x-axis and the corresponding longer ellipse axis to determine finger orientations as exemplarily sketched in ► Figure 3.19 (for only one contact area and contour).

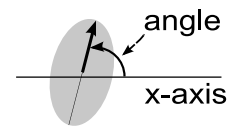


Figure 3.19: Finger contact area and its angle to the x-axis.

However, this approach suffers from inherent weaknesses due to its reliance on only the contour of a finger contact. For instance, it produces ambiguous results for circular contours (cf. ► Figure 3.20) because the axes in a circle can point towards any direction. Furthermore, a detected finger orientation could be wrongly skewed by 180° , since the longer axis of an ellipse provides two possible directions.

180° ADJUSTMENT. With the aid of the raw camera image, we are able to resolve the finger orientation in case it is wrongly skewed by 180° . We detect this kind of ambiguity by following the longer ellipse axis in both directions while looking for the direction that shows up a non-finger pixel or a non-hand pixel at first. The thereby detected direction is the one where the fingertip ends. Thus, we adjust the afore detected finger orientation if it is wrongly skewed. Henceforth, we use the term "ellipse-method + 180-adjust" to denote this addition to the ellipse-method.

Adjusting wrongly skewed orientations by means of raw camera images.

Figure 3.21: A Diffuse Illumination raw image with finger contours marked by white pixels.

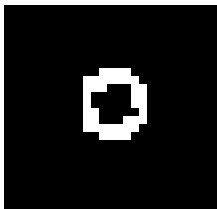


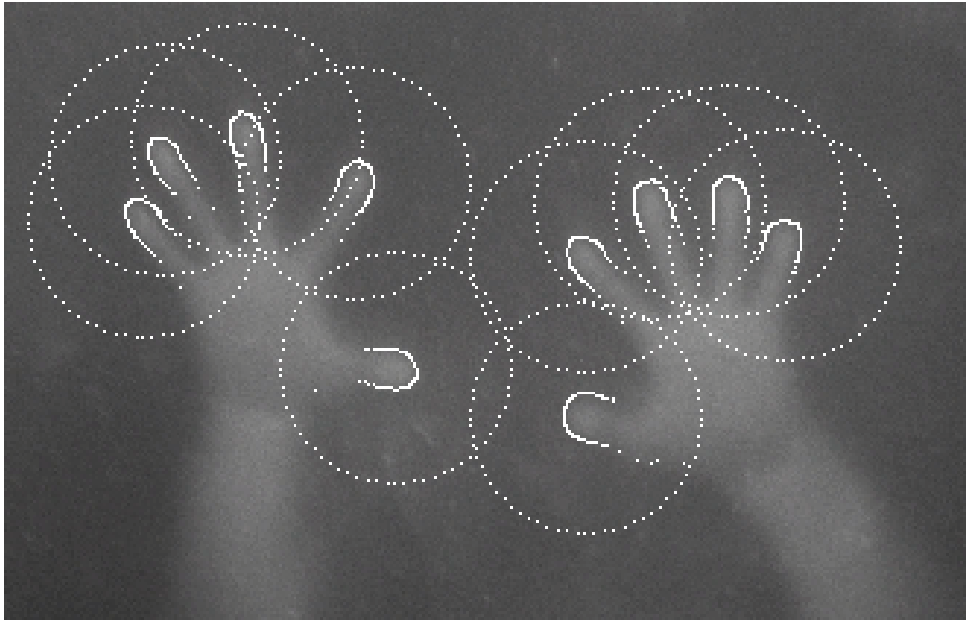
Figure 3.20: A small finger contact area that shows a circular shaped contour.

DISCUSSION. Ellipse-method and ellipse-method + 180-adjust represent a straightforward approach, but there are common cases in which touch interaction produces problematic contours for both methods. For instance, let us consider a child with small fingers who touch the surface, or finger touches from users who touch the surface only slightly or in a perpendicular manner. In such cases, the detected finger contour is very small and shows a circular shape in the worst case, as exemplarily depicted in ► Figure 3.20, which leads to imprecise or high deviant finger orientations. This is due to the following fact of small finger contours.

The fewer pixels that contribute to the contour, the more effect one pixel has on the detected finger orientation. This is even worse if we consider that camera noise always randomly affects pixels of the contour, which results in jumping values or jitter of the detected finger orientations. To overcome these issues and furthermore increase precision and stability of the detected finger orientations, we make use of the difference in brightness of proximate pixels provided by the corresponding raw images. We draw on that information to determine the finger contour and to derive the finger orientation from only a part of the finger contour.

A SIMPLE AND PRECISE DETERMINATION ALGORITHM. When considering images provided by Diffuse Illumination setups, we can identify each finger with its outer contour as depicted in ► Figure 3.21. What each finger contour has in common are two quasi axially symmetrical lines that converge circularly at the fingertip. Our approach exploits these symmetrical running lines to calculate finger orientations. For each finger, we are able to detect these two lines and they always point in the direction of the finger orientation. Hence, the identified finger orientation must be reliable.

Our approach exploits the symmetrical lines of finger contact contours.

**Figure 3.22:**

Points distributed on circles around the finger contact positions. Each point spans a path to the corresponding center coordinate of the finger contact.

Our algorithm is composed of three essential steps, which have to be applied consecutively in the given order to every detected finger contact position. Hence, detection of finger contact positions, which was previously described in ► Subsection 3.3.3, has to be done before or together with the detection of finger orientations.

Three consecutive steps for recognition of finger orientation:

1. Detect finger contour.
2. Determine symmetrical lines.
3. Determine angles in which the lines point to and average them.

1. DETECTION OF FINGER CONTOUR. As a first step, the outer finger contour for a given finger contact position has to be identified. This task is decisive for the remaining steps because the points of the contour intrinsically contribute to the precision of the finger orientation that is to be determined. We span a circle with the radius R (pixels) around the finger contact position and distribute points on that circle at an interval of 5° as sketched in ► Figure 3.22. The interval of 5° was sufficient for our image resolution to include almost all pixels on the finger contour. We have chosen $R = 40$ pixels for our images in order to cover at least two times the finger width of small people and one time and a half the finger width of people with chubby fingers. This value must be adjusted for other tabletop setups depending on the resolution of the raw images and the size of the tabletop surface. For higher resolution cameras, the amount of points on the circle might be increased for a higher precision of detection.

Distributing points on a circle around each finger contact position.

Search for non-contact pixels on paths to each point on the circles.

Result is a list of coordinates for each finger contact.

The next step is to perform a search for a contour pixel starting from the finger contact's center position to each point that was distributed on the circle. To be precise, we process 72 paths at an interval of 5° . Within each search run, we compare the pixel value of each point on the path with the pixel value of the center position. Once the difference of their values exceeds a certain threshold, the search terminates and the pixel coordinate on the path is noted in a list. For our implementation, we used a value of 18 for the threshold. This threshold may vary for other DI settings, depending on brightness and contrast of the captured images. Here, adaptive threshold calibration based on a histogram of the image could compensate for environments with varying light conditions. The result of this step is a list, which denotes whether a contour point was found or not, for each of the 72 points on the circle. In particular, the list meets the following constraints:

Definition 3. *Constraints of a contour-point list:*

1. *The ordered entries enumerated from 1 to 72 correspond to the points on the circle from 0° to 355° at an interval of 5° .*
2. *Each entry contains the (x, y) coordinate of a contour pixel.*
3. *If no contour pixel was found, then the entry's value is $(-1, -1)$.*

In theory, this procedure guarantees that the detected finger contour contains only the contour of the finger that we actually consider and not a contour point of an adjacent finger. Since we start the search from the finger contact's center position, the first pixel that terminates the search must belong to the same finger. Furthermore, most of the searches terminate quickly because the distances between the finger contour and the finger contact's center position are usually short.

2. DETERMINE SYMMETRICAL LINES. The following step operates only on the contour list (defined in the previous section) and determines the two quasi-symmetrical lines that contribute to the finger orientation.

All following steps operate only on the determined lists (arrays), which are treated as ringbuffers.

In what follows, the list is treated as a ringbuffer, that is, the next entry of the last entry with index 72 points to the first entry with index 1 and vice versa (for the previous entry of the first entry). Furthermore, the names in italic type (e.g., c_{start}) denote indexes into the list, corresponding to the usage in ► Figure 3.23 unless otherwise noted. The name *list* denotes the afore defined list of points from which a single point can be retrieved by means of square brackets (e.g., $list[c_{start}]_x$) as used in the programming language C.

At first, we perform a search for the largest range that contains only values of $(-1, -1)$. This range is defined as the gate $g_{start}, \dots, g_{end}$ and represents the part where the finger is connected to the hand as depicted in ► Figure 3.23. The complementary range $c_{start} = g_{end} + 1, \dots, c_{end} = g_{start} - 1$ is defined as the finger contour.

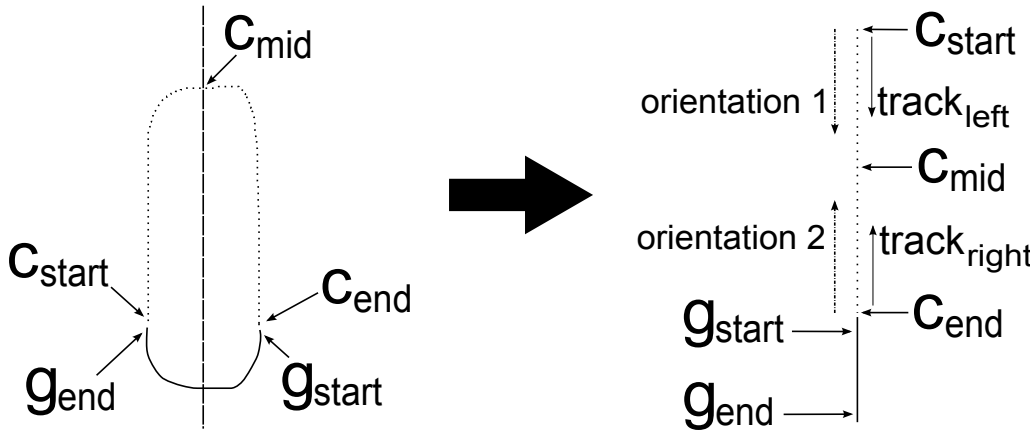


Figure 3.23: Finger contour points/locations in a ringbuffer with start and end points (left). List of contour points and its allocation (right).

When considering the finger contour as two parts with $c_{mid} : c_{start} < c_{mid} < c_{end}$, there are two contour tracks that have to be truncated in order to remove the part representing the fingertip. The fingertip part can only be used for detection of finger orientation if the contour tracks would be absolutely symmetrical to each other. This cannot be guaranteed due to different finger or hand postures, which may destroy the symmetry of the fingertip contour. Therefore, the fingertip part of the contour has to be removed as far as possible. For 5° intervals, we empirically determined that considering only 60% of the points in each contour track suffices to remove the fingertip points. That is, the number of points to include in each contour track is defined as $amount_c = 0.6 * (c_{end} - c_{start}) / 2$. This can be used independent of image resolution because a higher resolution would result in more contour pixels, but the relationship between finger and fingertip remains the same. Considering only 60% of the finger contour is a trade-off between including the contour that contributes to the finger orientation and omitting the fingertip contour, which would decrease the precision of detection.

Remove fingertip coordinates from lists.

For the following steps, we further define the left contour track as $track_{left} = c_{start}, \dots, c_{mid} - amount_c$ and the right contour track as $track_{right} = c_{end}, \dots, c_{mid} + amount_c$, cf. ► Figure 3.23. We also have to handle a rare case in this step. If the list does not contain a gate range, then we have an elliptical finger contour as exemplified in ► Figure 3.24. This case rarely occurs and is caused by low contrast of the raw image, which leads to falsely detected contour points stored in the list. In such a case, we skip the next step and proceed with the ellipse-method + 180-adjust to detect finger orientation, which was described in ► Subsection 3.3.4.1.

Skip lists that represent a completely connected ellipse.

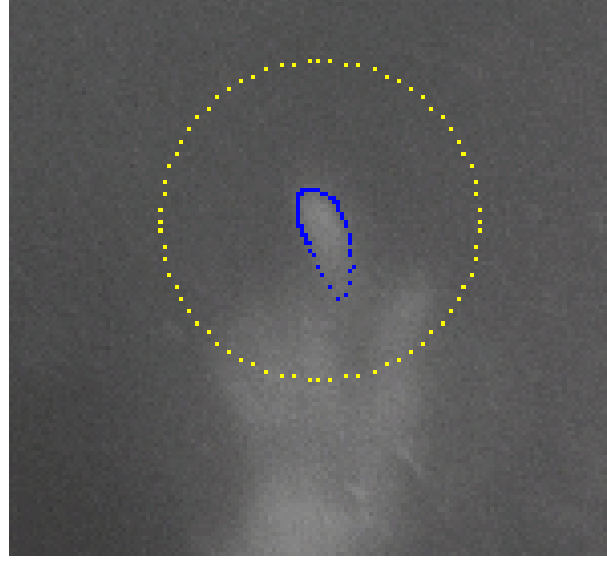


Figure 3.24: Example of an elliptical closed finger contour that was detected in a low contrast image. No gate range can be determined.

3. DETERMINE AVERAGE ANGLE. Both afore ascertained contour tracks of a finger contact contribute to the finger orientation based on the constraints defined in ► Definition 3. The constraints imply that c_{start} represents the start point of the left contour track and c_{end} represents the start point of the right contour track. Because of that, we also know that the tracks $track_{left}$ and $track_{right}$ point in the direction in which the finger is pointing to as well. Therefore, the last step serves to calculate the angles to the x-axis for $track_{left}$ and $track_{right}$, and average them afterwards in order to determine the final finger orientation.

► Definition 3
implicitly
disambiguates finger
orientations.

Angle computation
using linear
regression is more
precise but
computationally more
complex.

Angle computation
based on aggregated
slope values is much
faster.

For this last step, two methods with different complexity may be applied. The first one is a computationally complex method, which makes use of linear regression. The points of the contour tracks are considered as point clouds that serve as input for the computations. For each contour track, a linear regression model is fitted on its point cloud by means of least squares. The resulting slope of the regression line contributes to the finger orientation. Due to the complexity of linear regression calculations, this method is applicable for time-critical conditions only if sufficient processing power is available. A much simpler and faster approach that requires fewer computing power draws on the slopes for each point of a contour track. For this approach, the coordinate denoted by the index c_{start} or c_{end} is defined as the anchor point, depending on the contour track to work on. That is, either $anchor_x = list[c_{start}]_x, anchor_y = list[c_{start}]_y$ if we consider $track_{left}$ and otherwise $anchor_x = list[c_{end}]_x, anchor_y = list[c_{end}]_y$. The following instructions have to be applied to $track_{left}$ and $track_{right}$ independently. At first, the differences in x-value and y-value from the anchor point to all other contour track points are summed up. The resulting sum of x-values and y-values are then considered as the slope of the line that runs through the

point cloud of the corresponding contour track. This is formally specified with ► Equation 3.1, 3.2, and 3.3 where t_{start} and t_{end} are the indexes into the contour point list of the corresponding contour track. Finally, both slopes have to be averaged to determine the final finger orientation.

$$(3.1) \quad sum_{\Delta x} = \sum_{i=t_{start}}^{t_{end}} list[i]_x - anchor_x$$

$$(3.2) \quad sum_{\Delta y} = \sum_{i=t_{start}}^{t_{end}} list[i]_y - anchor_y$$

$$(3.3) \quad slope = \frac{sum_{\Delta y}}{sum_{\Delta x}}$$

This simpler approach admittedly is prone to errors due to wrong initialization of the anchor point. However, the evaluation results that we present in ► Subsection 3.3.6 show that this simple method still enables recognition rates that are quite close to those of the linear regression method.

3.3.5 Recognition Rates

In order to obtain recognition rates that have relevance to real data and that would occur during real tabletop interaction, we employed a Diffuse Illumination setup and captured a raw video stream from an infrared camera that was mounted in the tabletop and was directed towards the tabletop surface. The video shows the hands and fingers of a user who touches the surface the same way as he would do to interact with an application. Thereby, he uses combinations of one hand and two hands and with different finger combinations multiple times. For example, the finger combinations included combinations used to move objects with the index finger (► Figure 3.25a) or multiple fingers (► Figure 3.25b), grasping with all fingers (► Figure 3.25c) or zooming with fingers of one hand (► Figure 3.26a) or both hands (► Figure 3.26b). During recording of the video, the tabletop showed a black screen and gave no visual feedback. Furthermore, the fingers in some of the captured images were blurry due to quick continuous hand and finger movements. Hence, the video consists of images that occur in multi-touch tabletop setups during realistic scenarios.

Recognition rates based on a video stream from a Diffuse Illumination setup with finger interaction and gestures of realistic scenarios.

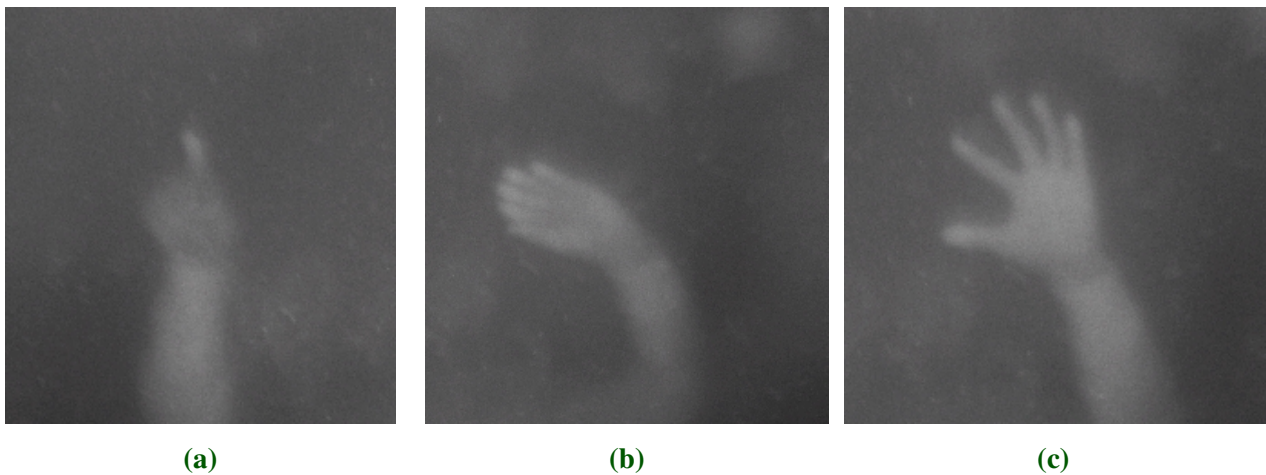


Figure 3.25: Finger gestures and combinations used for evaluation: (a) Move an object with the index finger; (b) Move an object with multiple fingers; (c) Grasping with all fingers.

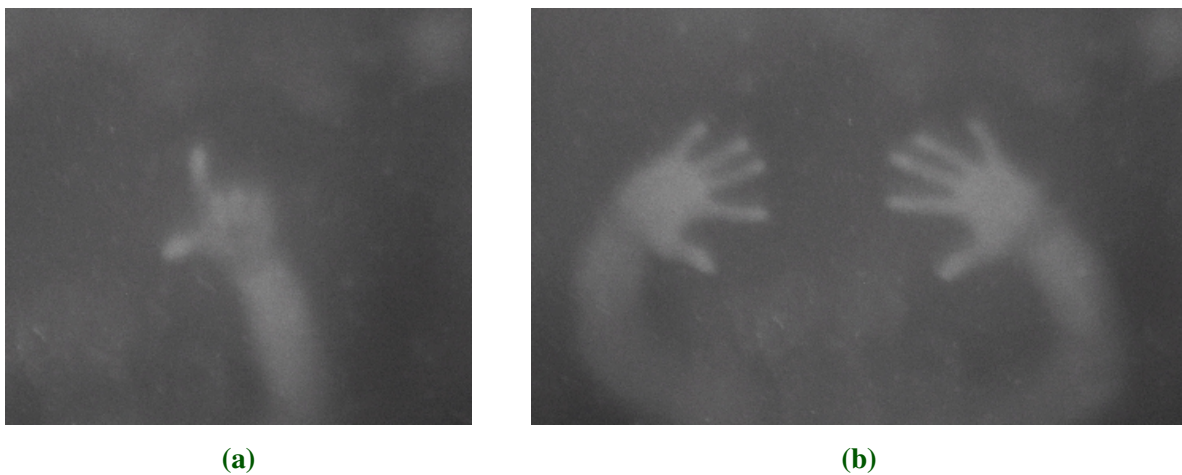


Figure 3.26: Finger gestures and combinations used for evaluation: (a) Zoom gesture with fingers of one hand, (b) zoom gesture with fingers of both hands.

In addition to the video, we created reference finger orientations for each frame of the video. These reference orientations were then used to evaluate the accuracy of the detected finger orientations. In contrast to the work of Wang et al. [2009], we chose a different method to evaluate the performance of our approach. In Wang et al. [2009], the authors evaluated their approach by investigating to what extent the system's response was in line with the user's objectives when conducting a set of predefined tasks with the index finger. In the present work, we propose a different evaluation approach that compares the finger orientation found by our algorithm with the finger orientation perceived and determined by three independent human judges. The advantage of our method is that it enables a task-independent evaluation and gives a measure of the precision that is more focused on continuous interaction.

Evaluation of
performance is
different from Wang
et al. [2009].

3.3.5.1 Reference Orientations

The video for the reference finger orientations was available in an uncompressed format to preserve the raw data and consisted of 749 frames captured at 30 fps with an image resolution of 640 x 480 pixels. The resolution covered a physical surface space of 60 x 80 cm. For creating the reference, each frame was converted to a blob image and the center position of each recognized blob was stored with its frame number in an XML-file. In all, 2007 finger contacts were detected. To ease the annotation of finger orientations and to conveniently generate the reference, we developed a graphical tool. The tool presented each raw image frame along with the position of the detected finger contacts to the annotator. The annotator then had to manually adjust the finger orientations, which were previously detected through the ellipse-method + 180-adjust approach.

Database with reference orientations was built using a graphical tool.

Overall, three persons annotated 6021 finger orientations. Hence, three finger orientations were collected for each finger contact. The annotators were instructed to repeat an adjustment as often as required, if they were not absolutely sure about the correctness of the visually perceived finger orientation and their adjusted finger orientation. The data gained through the annotations were used to build the reference finger orientations by calculating an average finger orientation for each finger contact from the annotated data.

Database with reference orientations was created by three annotators.

3.3.5.2 Accuracy of the Reference Orientations

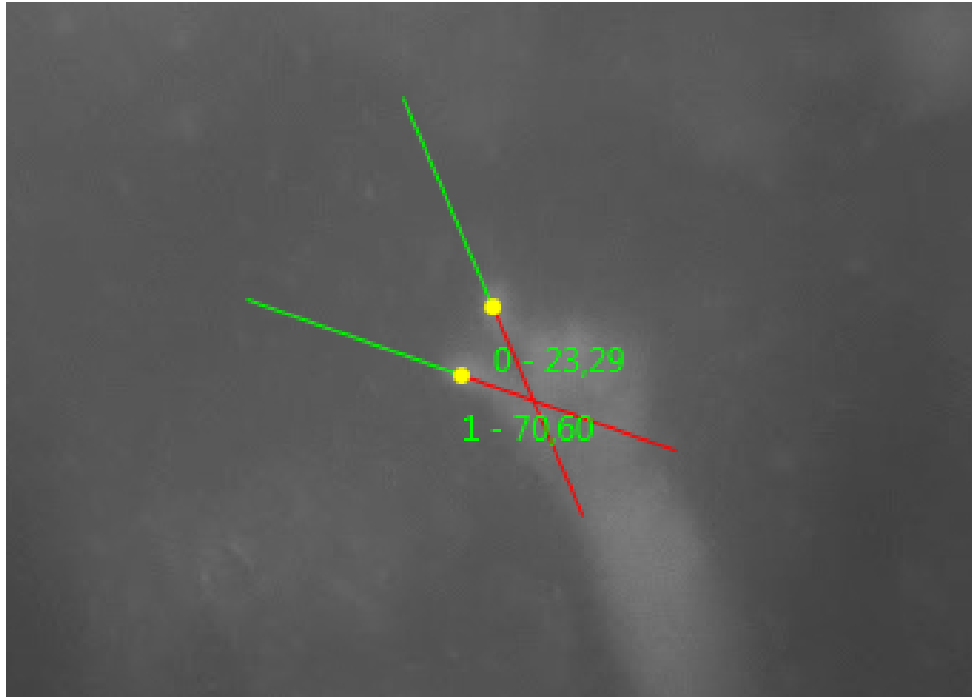
The reason why we averaged annotations obtained from three different persons is that different persons may determine slightly different finger orientations due to camera noise, image contrast, and image resolution. Our tool enabled the annotators to adjust the finger orientation angle with a precision of two decimal points as exemplarily depicted in ► Figure 3.27.

► Figure 3.27 also shows that the raw images are blurry and have low contrast. These facts make it difficult to pinpoint a definite finger orientation. Therefore, the annotation tool provided a colored visual (ruler alike) line, which assisted the annotators in adjusting a proper finger orientation. By means of the visual line, annotators were able to "extend the particular finger" into both directions and provide a more precise finger orientation value. This process enabled us to obtain data from multiple annotators and take the averages for the reference database. Overall, the reference orientations had a standard deviation of 3.34°.

Visual ruler assisted the annotators in order to maximize the accuracy of the database.

Figure 3.27:

Annotated index-finger and thumb of the same hand that perform a gesture on the surface. Green/red colored rulers help annotators determine the finger orientation.



Method	< 5°	< 10°	< 15°	< 25°	SD
1	41.5%	60.49%	68.81%	74.39%	±86.46°
2	49.58%	74.34%	84.65%	92.53%	±23.19°
3	75.29%	93.02%	96.86%	99.3%	±6.17°
4	75.24%	94.87%	97.81%	98.9%	±6.43°

Table 3.2: Recognition rates and standard deviation for four methods:

1 = ellipse-method, 2 = ellipse-method + 180-adjust,

3 = Contourtrack + simple slope, 4 = Contourtrack + regression line.

3.3.6 Results

The precision of our approach is given in ► Table 3.2, which provides a comparison between the recognition rates of our approach with the recognition rates of the ellipse-method and the ellipse-method + 180-adjust. As for our approach, we implemented both variants described in ► Subsection 3.3.4.1 for finding a line that runs through the contour track. In the following, the term "Contourtrack" denotes our approach. ► Table 3.2 shows the recognition rates for all four approaches and their standard deviation from the finger orientation in the reference. The table lists the recognition rates for four tolerance values: 5°, 10°, 15°, and 25°. All recognition rates in the 5° column show the percentage of recognized finger orientations at which the difference to the reference is less than 5°. The same interpretation applies to the 10°, 15°, and 25° column.

We implemented all discussed approaches for comparison.

We evaluated with four different tolerance levels.

The naive ellipse-method shows the lowest recognition rates (60.49% at a tolerance of 10°) and a high standard deviation of 86.46° , which were mainly caused by wrongly twisted (by 180°) finger orientations. This method hugely benefits from correction of the wrongly 180° twisted finger orientations (ellipse-method + 180-adjust) as shown in the second row of ► Table 3.2. The (180-adjust) extension improved the recognition rates (74.34% at a tolerance of 10°) and reduced the standard deviation drastically to 23.19° . However, the recognition rates of the naive methods ellipse-method and ellipse method + 180-adjust show that their precision is not reliable enough to be used in realistic applications. In the third row and the fourth row, the recognition rates of our new approach are shown, which are much better than the naive ellipse methods. The values of both show good results, which exceed 93% for a maximum tolerance of 10° . If we accept a maximum tolerance of 25° , then both methods provide recognition rates over 98%. Both standard deviations are less or equal 6.43° , which is only 3.09° worse than the standard deviation in the human reference data with 3.34° .

Adjustment of wrongly skewed finger orientations greatly improved the naive approach.

Our approach exceeds 93% for 10° tolerance and 98% for 23° tolerance.

3.3.7 Discussion

The precision of our approach increases with higher image resolution and image contrast. The more pixels that can be used for a finger contour, the better the precision. The better the image contrast, the better a finger contour can be detected. Pre-processing steps to reduce image noise or to improve image contrast would further improve the performance, but we haven't included such steps so as to keep the approach simple and fast. By this way, the approach can be used as a lightweight add-on to already established tracker applications.

3.4 Hand Distinction

Contribution Statement: In this section, we contribute to enhancements of gesture recognition approaches by explicitly distinguishing one-handed from two-handed gestures based on the finger input property hand distinction. Tabletop tracking components such as used in the Microsoft Surface/PixelSense tabletops provide a lot of finger input properties to applications but no hand related information. We approach this gap with an algorithm that maps touch contact areas to their corresponding hands with a high level of precision. In addition to that, we discuss observations from an empirical evaluation of the algorithm. An earlier version of this section's content has been published as a peer-reviewed full paper [Dang et al., 2009] at the ITS (Interactive Tabletops and Surfaces) 2009 conference. <https://doi.org/10.1145/1731903.1731925>

Definition of hand distinction.

Hand distinction names a computational process that determines whether a set of touch contacts originates from one hand or from different hands. In the latter case, hand distinction maps each touch contact (of a set of touch contacts) to a hand (of a set of hands). This is particularly useful for multi-touch tabletops where user input can potentially originate from different hands of one or several users and from multiple fingers of their hands. Hand distinction in sensing systems provides the corresponding hand for each touch contact that was detected on the sensor surface. In the following, we denote this information as hand distinction, which is a finger input property consisting of a numeric value. Each hand distinction value represents a human hand located above the tabletop surface and each touch contact is mapped to one of the hand distinction values.

A hand distinction value is a numeric value that represents a human hand.

Sensor categories that enable hand distinction.

As can be seen in the overview in ► Table 3.1, a hand distinction value can be provided by sensing technologies of the category DI and Depth due to the rich information within the raw images. In such images, the corresponding hands of touch contacts are directly recognizable and hence can be detected through computer-vision approaches. For example, the tabletop tracking application of MultiTaction cells¹⁵ provides a so-called *handId*, which identifies the hand to which a finger contact area belongs to. The sensing category FTIR enables detection of hand distinction for FTIR variants such as shadow tracking [Echtler et al., 2008, Iacolina et al., 2011]. Classic FTIR tabletops cannot directly determine hand distinction values as a finger input property. The last sensing category

¹⁵<http://www.multitaction.com>

Capacitance also lacks support for hand distinction as a finger input property. In order to fill this gap, we present an approach to derive hand distinction values based on two other finger input properties that are available or derivable by every other sensing category, namely the center position and the finger orientation. Our approach can also be applied to tabletop systems that do not allow modification of the tracking software but do provide both required finger input properties.

We provide an approach to enable hand distinction as a derived finger input property.

3.4.1 Usage of Hand Distinction

Applications that are able to determine whether touch contacts are provided by one or several hands may exploit the potential of a more natural and richer repertoire of input gestures. Often, a repertoire of gestures can only be extended at the expense of lower robustness, since new gestures are distinguished from existing ones only by subtle variations. The inclusion of two-handed gestures would, however, not result in a significant loss of accuracy because information on handedness may be employed as a highly discriminative feature to classify input. In applications that emulate the functions of mouse buttons, techniques to distinguish one-handed from two-handed input could, for example, ensure that mouse actions are only triggered if touch contacts come from one hand and not from different hands that accidentally form a similar constellation.

Hand distinction enables extension of tabletop gestures.

Moscovich and Hughes [2008] presented a study, which demonstrates that one-handed input and two-handed input is not interchangeable, since handedness has an impact on the ease and accuracy with which gestures may be conducted. In particular, they showed that one-handed input is suitable for moving, stretching or rotating an object while two hands are more decent for tasks wherein separate control of points, such as selecting a region, is required. Their study advocates the use of both uni-manual and bi-manual gestures depending on the accuracy with which certain tasks can be performed. Here, distinguishing hands helps adapt the application behavior to offer possibilities to ease precise tasks in case of two hands, for example, by changing the resolution or speed of object movements.

Hand distinction enables distinguishing one-handed from two-handed gestures.

Even though many research projects exploit the potential of a richer repertoire of input gestures resulting from two-handed input, current works usually do not consider cases where a collection of touch contacts should be interpreted differently depending on whether they originate from the finger of one hand or from the fingers of several hands. For example, Benko et al. [2006] presented a dual finger stretch technique where one finger is used to select an area of the interface and a second one is used to scale the area (cf. ► Figure 3.28b). However, when evaluating their gestures, they start from the assumption that all gestures are executed with two fingers of two different hands belonging to a particular

Hand distinction allows different actions depending on whether gestures are performed by one or more hands.

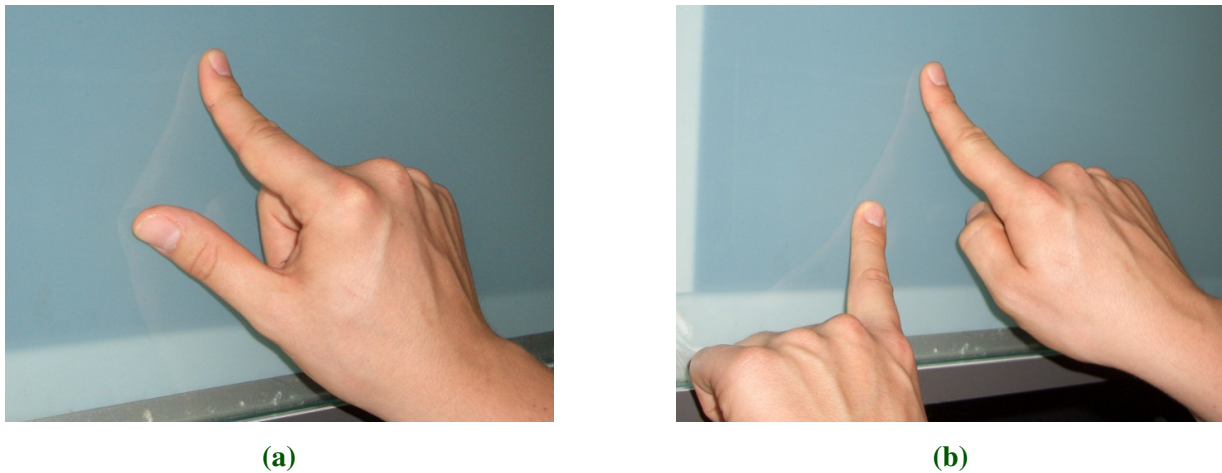


Figure 3.28: An example for ambiguous recognition if no hand distinction is available. Gestures with one hand (a) and two hands (b) creating the same touch positions and tracks.

person. Such a gesture could also be performed by two fingers of the same hand as depicted in ► Figure 3.28a. As a consequence, it might be difficult to distinguish a dual finger stretch gesture performed with the fingers of the same hand from a dual hand duplicate gesture where a copy of a visual object is produced by pulling the object apart with the fingers of two separate hands. Another example pertains to the application behavior for multiple objects when two objects are moved together. A one-handed gesture could result in merging those objects, whereas a two-handed gesture would overlay one object with the other object for a precise comparison task.

Peltonen et al. [2008] reported on problems that arise when multiple users interact in parallel on large surfaces and unintentionally break territorial boundaries. Techniques that distinguish between one-handed and bi-manual input may help resolve such conflicts. Consider, for example, the case where a photo is accidentally zoomed because two users are trying to move it toward themselves as depicted in ► Figure 3.29. If a system is, however, able to recognize that two users are manipulating one and the same object in parallel, but with different intentions, it might respond to the users' behavior in a more appropriate manner. For example, it might lock the object until the users have agreed upon what to do with the object or create a duplicate of the object.

Terrenghi et al. [2007] conducted a study to investigate whether people manipulate physical objects on a tabletop differently than digital objects. They found that users tended to use just one hand when interacting with digital objects. Distinguishing hands and offering interaction based on handedness might encourage users to use both hands. For example, participants were requested to spatially structure objects. In the physical tasks, users could move multiple pieces all together to form a structure, but it was hard to imitate such an action in the digital tasks. However, hand distinction could enable an adoption of

Hand distinction
allows
avoiding/resolving
conflicts in
collaborative tabletop
work.

Hand distinction might
help motivate
bi-manual interaction.

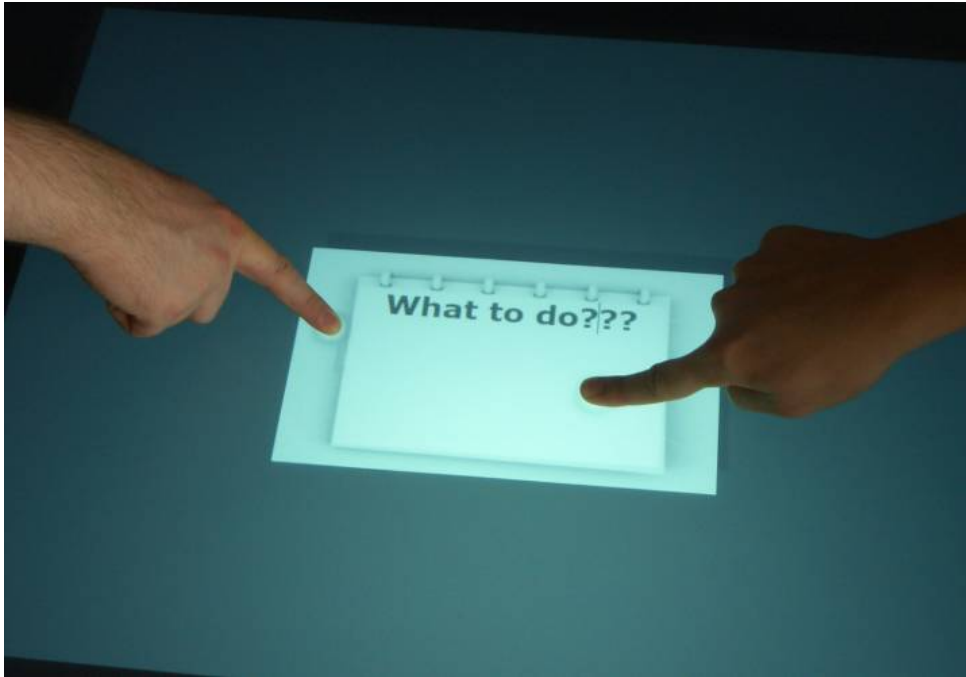


Figure 3.29: Example for conflicts in collaborative tabletop interaction: two different users who perform a pulling away gesture on the same object and at the same time.

the application behavior, so as to select all underlying and overlying objects for movement if both hands are touching an object, whereas one hand selects only the directly touched object.

3.4.2 Related Work

Various attempts have been made to distinguish one-handed touches from two-handed touches. An early example can be found in the DiamondTouch [Dietz and Leigh, 2001] tabletop, which makes use of modulated electric fields. When a user touches the surface of the table, the contact areas are capacitively coupled through the user to a receiver, corresponding to that user. By this way, the DiamondTouch table is able to determine for each touch point to which user it belongs, cf. ► Subsection 3.1.2.2. However, the DiamondTouch tabletop is not able to distinguish between one-handed and two-handed gestures from one and the same user.

DiamondTouch tabletops map touch contacts to users, thus distinguish between hands of different users but not hands of the same user.

Echtler et al. [2008] presented an approach for FTIR-based tabletops that features the possibility to map finger touches to the hands of a single or several users. It is based on a top-mounted infrared light source that lets the hands and arms throw shadows onto the surface. However, they haven't explored their setup to explicitly distinguish hands yet. Another approach was employed by Dohse et al. [2008] who enhanced the interaction on an FTIR-based tabletop with a camera placed above the interactive surface. This technology allowed them to track hands by means of computer-vision techniques, thus assign a specific touch contact to a user or his hand.

FTIR extensions for hand distinction.

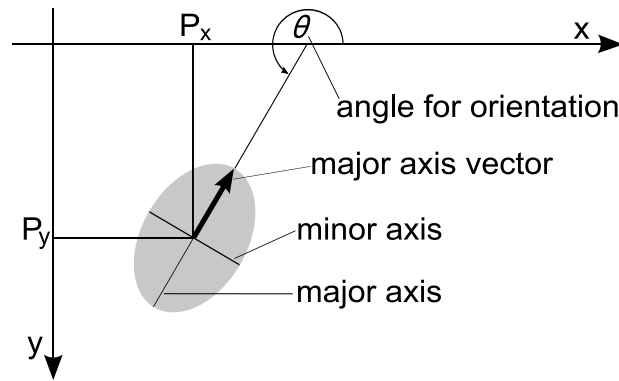


Figure 3.30: A finger blob represented as an ellipse with position and angle drawn into a Cartesian coordinate system.

Auxiliary cameras also enable hand distinction.

Malik and Laszlo [2004] and Do-Lenh et al. [2009] presented various one-handed and two-handed input techniques for a visual touchpad and an augmented tabletop. Both works distinguish single gestures based on positional information of the fingertips. In their case, there is no need to disambiguate gestures based on handedness because the chosen gestures are sufficiently different. Nevertheless, their approaches to determine finger-features can also be used to distinguish between one-handed and two-handed input on infrared images captured by tabletop setups.

3.4.3 Center Position and Finger Orientation

The idea behind our approach exploits information on the anatomy of human hands and fingers to distinguish between one-handed and two-handed gestures. We will show that only two finger input properties, namely the center position and finger orientation, are sufficient to reliably map finger contact areas to hands and ultimately to derive hand distinction values.

► Figure 3.30 shows a sketch of the finger input properties in the geometric arrangement and denotations that are used throughout this section. The finger contact area is approximated through a (gray colored) ellipse with a position (P_x, P_y) and the lengths of the major and minor axis. The major axis vector of the ellipse allows us to calculate the angle of x-axis, which represents the orientation of the finger contact area. ► Figure 3.30 assumes that the position $(0, 0)$ is located at the top-left-corner and that the y-coordinates increase downwards as commonly used by most operating systems.

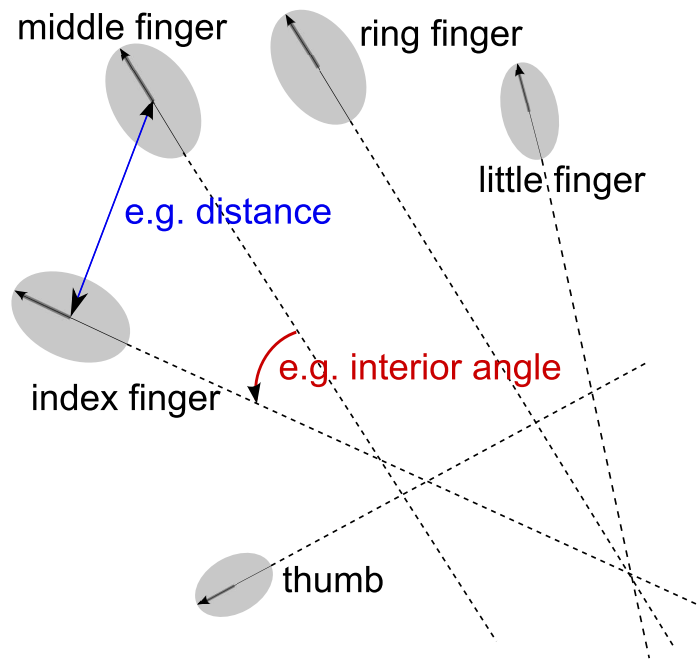


Figure 3.31: Contact areas of the five fingers of a hand with the intersection points of its backward oriented lines (black colored). Distance between center locations (blue colored). An interior angle between oriented lines (red colored).

3.4.4 Concept for Finger Mapping

Our objective is to map a set of finger contacts to a set of corresponding hands. Each hand is represented by a unique identifier called *handId*, that is, the finger contacts are assigned to a *handId* based on the fingers' associated joints to a hand. To achieve this task, we take advantage of the fact that the constellations of touch contacts that belong to the same hand are naturally limited. In the following, we present a parameterized heuristic approach to classify finger contacts to a *handId* based on such limitations. We identified two features that are available as finger input properties and are sufficient to achieve correct assignments for most of the cases. The natural limitations of finger constellations are formulated as constraints on these features, which can be applied efficiently with our heuristic.

Our finger-mapping concept exploits natural limitations of finger constellations formulated through constraints.

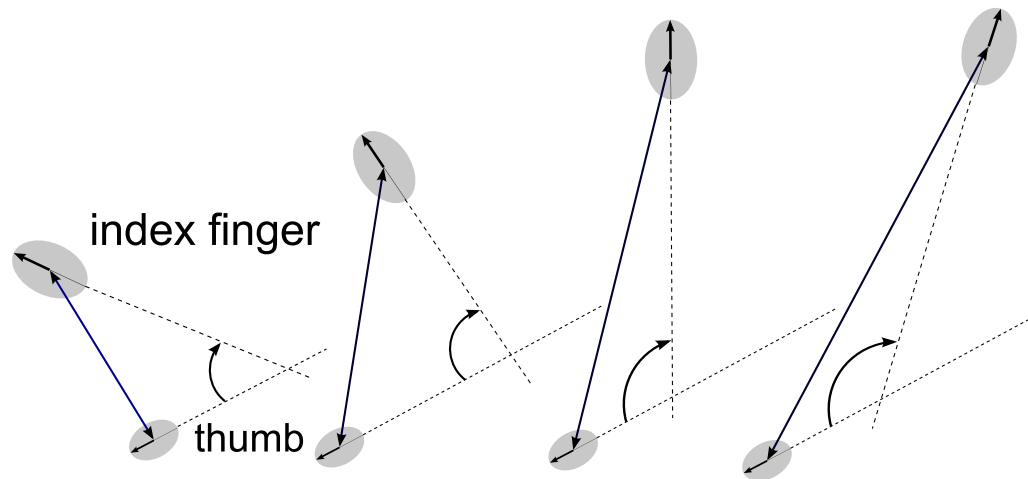
3.4.4.1 Analysis of Finger Arrangements

When considering finger arrangements that are possible with human hands, we take only arrangements into account that can actually be formed for touching a flat surface. ► Figure 3.31 exemplarily depicts the layout of the thumb and all fingers of a hand with its oriented lines. The oriented lines are the dashed lines elongated into the opposite direction of the finger orientations. If looking at only two contact areas at a time, we can identify the distance between their center

Only finger arrangements for touching a flat surface are considered.

Figure 3.32:

Variation in distance and angle of thumb to a finger. Combinations are from left with low distance and acute angle to right with far distance and obtuse angle.



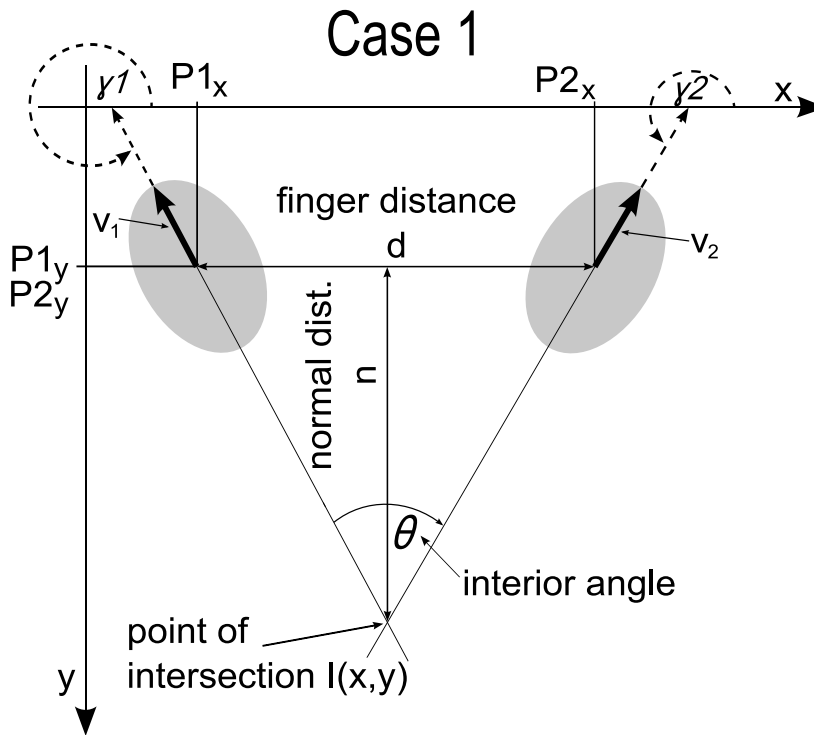
positions, the course of their oriented lines and the interior angle between their oriented lines. Any other finger contact composition of one hand can be looked upon as a subset of the depicted contacts. The relative orientation of finger contacts to each other varies just a little due to the anatomy of the human hand skeleton. That is, normally for two fingers abreast, the distance between their contact positions can reach a maximum value. The thumb represents an exception in terms of the maximum distance. In combination with a finger, the thumb enables a higher distance between their center positions and a higher interior angle between their oriented lines as sketched in ► Figure 3.32. Upon these observations, we deduce constraints based on geometrical properties derived from center positions and finger orientations.

3.4.4.2 Constraints and Conditions

Reduction of hand distinction to comparing of only two finger contact areas.

Calculation of base values from two finger contact areas.

The task of mapping a set of fingers to a set of hands can be decomposed to sub-tasks, which compare only two finger contact areas for each contact area with each other in the set. Let's assume that the set of fingers contains n finger contacts, then $(\frac{n^2}{2} - n)$ sub-tasks have to be performed. These sub-tasks can be processed in parallel to reduce latency. ► Figure 3.33 delineates such a comparison and the corresponding geometrical properties. Typically, the center positions $P1(x, y), P2(x, y)$ and the orientations as angle of x-axis γ_1, γ_2 are given, which enable us to calculate the distance d between the center positions, the interior angle Θ between the oriented lines, the intersection point $I(x, y)$ of the oriented lines, and the distance n between the intersection point $I(x, y)$ and line d by means of the normal line. Furthermore, we have to calculate the pair of gradient vectors v_1, v_2 using the given orientation angles, which are definite and unambiguous as compared to the ambiguity of slope values (i.e., a slope value may theoretically increase to ∞).

**Figure 3.33:**

Oriented lines of two finger contact areas with intersection point in the back together with the distance between two fingers and the normal distance to the intersection point.

CONDITION 1: DISTANCE AND PROXIMITY. The distance between two touch contacts is treated as a first indicator for being from the same hand or not. For this purpose, we empirically determined a maximum distance $D_{max} = 10,55''$ that limits the largest distance that a thumb and one of the other fingers can reach. If the distance d exceeds D_{max} , then the touch contacts are from different hands. We also empirically determined a distance $D_{adj} = 3,5''$ for adjacent fingers where the distance of the touch contacts is less than D_{adj} as can be seen in ► Figure 3.31.

Distance between two finger contact areas as a first indicator.

CONDITION 2.1: INTERSECTION POINT. The next step considers the intersection point $I(x, y)$, which has to be located behind both corresponding touch contacts. This condition can be evaluated by means of the intersection point together with either both center points $P1(x, y), P2(x, y)$ or their gradient vectors v_1, v_2 . A special case occurs with parallel-oriented contact areas, which have no intersection point. This case has to be caught by means of the gradient vectors and will be handled with condition 2.2. ► Figure 3.34 illustrates a situation in Case 2, where the intersection point is located in front of both touch contacts, thus violates the intersection point condition twice. This arrangement cannot be produced by fingers of one hand, whereas the thumb in combination with another finger of the same hand is able to create such an arrangement. To handle this case, we employ temporal information that make use of previously assigned handIds to the touch contacts. For instance, when the thumb creates contact areas such as in Case 2, it may perform a grab gesture on the surface that usually starts with correctly assigned handIds for the touch contacts. Constellations such

Parallel oriented contact areas have no intersection point.

Exploit previously mapped hand-ids to stabilize Condition 2.1.

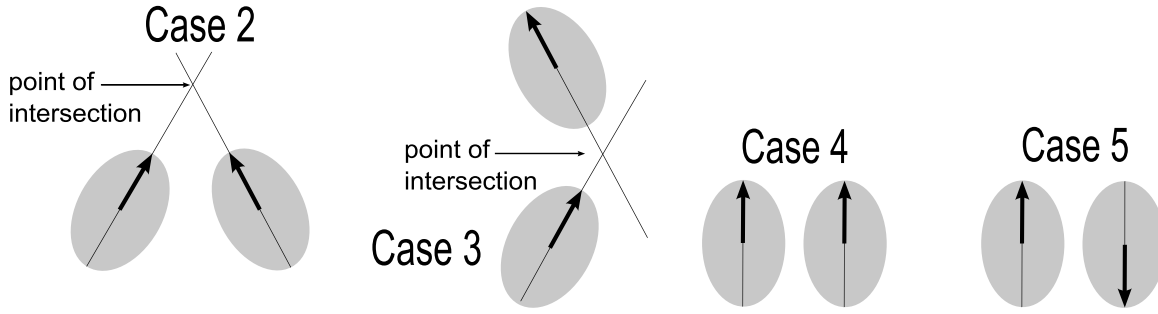


Figure 3.34: Case 2 and 3: Intersection point in front of at least one line; Case 4: Quasi parallel-oriented finger; Case 5: Quasi-reverse oriented finger.

as in Case 2 are then fused with the previously assigned handIds of touch contacts and hence get assigned correctly. Case 3 in ► Figure 3.34 also violates the condition because the intersection point is in front of one of the touch contacts. Such an arrangement can be produced, for example, through the pump gesture as proposed in [Sousa and Matsumoto, 2007]. The pump gesture is performed by describing an arc movement with the thumb under another finger. As for Case 2, this case can be handled by considering temporal information.

CONDITION 2.2: ADJACENCY AND PARALLELS. Adjacent touch contacts can be arranged in parallel or quasi-parallel as sketched in Case 4 of ► Figure 3.34. In these cases, the interior angle between the oriented lines is very small and enables us to stabilize adjacent contacts. Therefore, we define an angle Θ_{adj} and assume that two touch contacts belong to two adjacent fingers if their distance d is less than D_{adj} (distance condition) and the interior angle Θ is less than Θ_{adj} . We found 45° to be a reasonable value for Θ_{adj} . Not directly adjacent fingers, for instance, the index finger and the ring finger, can also be placed in parallel. For such cases, the assignment depends on the value of D_{adj} , which is adjusted to include a proximity of 3 fingers. The touch contacts in Case 5, however, cannot be produced by only one hand during normal touch interaction. Hence, Case 5 results in finger contacts of different hands by examining the gradient vectors.

CONDITION 3: DISTANCE RELATIONSHIP. Condition 1 and 2.2 are sufficient to recognize adjacent finger touches. However, condition 1 and 2.1 are not sufficient because Case 1 in ► Figure 3.33 for distances $D_{adj} < d < D_{max}$ can still be formed by fingers of two different hands. Therefore, we adopt the limitations imposed by the limbs and joints of human hands to deal with that situation. ► Figure 3.35 depicts the distance between two touch contacts and the normal distance to their intersection point in order to show the following relationship. The more distant two touch contacts become, the more the interior angle between their oriented lines will increase and the shorter the corresponding normal

Determining adjacent fingers through distance and interior angle.

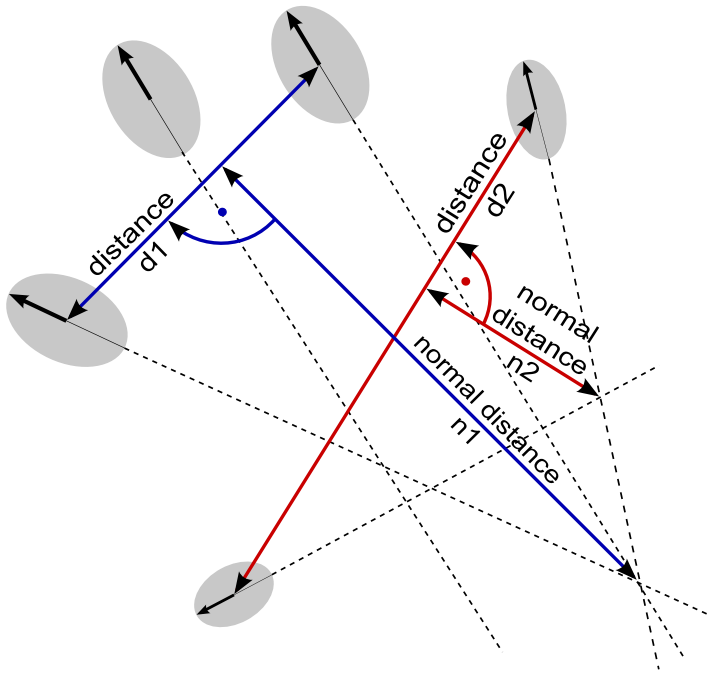


Figure 3.35: Two examples of the relationship between the distance of two fingers and the normal distance to the intersection point of its backward oriented lines.

distance becomes, for example, $d_1 < d_2, n_1 > n_2$ in ► Figure 3.35. Condition 3 exploits this relationship to constrain the constellation and distances of touch contacts, thus exclude or recognize touch contacts as being from the same hand. For this condition, we empirically determined a value $N_{max} = 14,06''$ for the maximum normal distance and allow only normal distances n that are lower than N_{max} . Normal distances higher than N_{max} are handled by condition 2.2 or indicate touch contacts from different hands.

$$(3.4) \quad score = (D_{max} - d) - \frac{n * D_{max}}{N_{max}}$$

In ► Equation 3.4, we scale the normal distance n to the proportion of the maximum distances D_{max} with N_{max} . Since the distance d is in inverse proportion to n , we subtract the result of the previous adaptation from the difference between D_{max} and d to get the *score*. If the *score* is positive, then the touch contacts are from the same hand. Otherwise, the *score* is negative and the intersection point is too far away. Noteworthy to mention is that ► Figure 3.35 shows touch contacts of one hand with the fingers outstretched, but gestures are composed of constellations in which fingers are placed close together as well. Our approach safely handles such constellations through the distance and adjacency condition.

3.4.4.3 Temporal Adaptation

Caching and reusing
previously mapped
handIds greatly
stabilize recognition.

Tracking systems usually assign an incremental finger identifier (numerical value) to every finger touch contact and keep track of the touch contact while retaining the finger identifier for the same finger. By examining frames over time using the finger identifier, we are able to capture and carry along dynamic hand-mapping information. For a new frame, the cached handIds are taken over from the previous frame. After that, the conditions are applied to the new frame so as to correct the adopted mapping or handle new touch contacts. To cope with situations where fingers of different hands are close to each other or entangled, our implementation maintains the number of fingers for each handId from the previous frame.

When two hands move close together, then their finger identifiers should have already been mapped to handIds in a previous frame. Therefore, we considered the number of fingers when deciding whether to change a handId or not. The temporal adaptation is an important component of our approach and improves classification enormously.

3.4.4.4 Diversity of Finger Anatomy

Adapting to small or
large fingers through
adjusting distance
thresholds D_{max} and
 D_{adj} .

The dimensions of hands and the extent of fingers vary between individuals as well as with increasing age, but the geometrical constraints as described with condition 3 and 2.1 still remain valid. Depending on the size of users' hands, the maximum distance D_{max} might be too small and has to be increased. On the contrary, the maximum distance for adjacent fingers D_{adj} could be too big for children and thus require an adaptation. We have not tested our approach for children. However, a possible approach to dynamically adjust the parameters is to take the physical mass of the considered finger touch contacts into account.

3.4.5 Empirical Evaluation

In order to evaluate our recognition approach, we conducted a user study to gather empirical data for finger touch contacts that originate from real touch interactions. This data serve to give proof of the recognition performance and to provide statistics of recognition rates.

3.4.5.1 Apparatus

For collecting finger contact data and evaluating our approach, we utilized a Microsoft Surface Developer device. It is a horizontal direct-touch tabletop that employs the DI sensing technology and provides reliable finger orientation for finger touch contacts. The tabletop device rear-projects images onto a surface measuring 24" x 18" with a resolution of 1024 x 768 pixels.

Evaluation employed a Microsoft Surface tabletop.

3.4.5.2 Experiment

The experiment aimed at collecting touch contact data of diverse hand dimensions so as to verify our finger-mapping concept and provide quantitative results. We even went one step further and tested the performance of our concept for conflict situations in which fingers of two hands crossed each other or were located closed to other hands.

PARTICIPANTS AND TOOLS. We recruited 10 volunteers for the experiment, 4 males and 6 females aged from 21 to 60. One of them was left-handed, and all of them, but two had already experience with multi-touch interaction through the Apple iPhone. We implemented a tool to record all finger touch contacts with their contact properties and timestamps, that is, the tool recorded and stored them into an XML-file. We also developed a tool called contact-player to step through every recorded frame and to visualize the contact data similar to a movie player that shows the flow of the touch contacts. The contact-player implemented a prototype of our approach to map finger contacts to hands and enabled annotators to mark falsely mapped finger contacts. After manually verifying all recorded frames, the contact-player calculated statistics of the number of fingers and frames together with correctly or falsely mapped fingers, so as to provide recognition rates. For these statistics, only frames with more than one touch contact were considered.

Evaluation tool is capable of recording and storing touch contacts as well as visualizing recorded touch contacts like a movie player.

METHODOLOGY. The experiment used a within-subject design and consisted of five tasks, each deliberately designed to verify the approach step by step. The first task validated the geometry assumptions for only one hand on the surface and for different hand dimensions. The second task did the same for two hands while one of the hands was located below the other on the surface. The third task addressed the geometry assumptions while the involved fingers exploited their scope to vary angle and distance. Tasks four and five served to simulate conflict situations with finger arrangements that are often used for zoom or grab gestures. All participants were told to approach the surface with their fingers as if they would manipulate objects during performance of the following tasks:

Data was collected through five tasks using a within-subject design experiment.

The basic task with 25 previously defined finger combinations.

Bi-manual basic task.

Basic task with maximizing possible extent of finger movements.

Simulate and reproduce bi-manual conflicts.

Simulate and reproduce multi-user conflicts.

1. They touched the surface with one hand and then with the other hand. For each hand, they used 25 defined finger combinations. The combinations consisted of fingers and thumb and were chosen to cover up arrangements that are frequently used to resize, move or grab objects.
2. They performed the same combinations as in task 1. but simultaneously with both hands.
3. They performed the same combinations as in task 1. and shrank up and stretched the involved fingers once to their possible extent.
4. They placed the following combinations with two hands apart from each other on the surface, moved the fingers close together, moved the fingers to cross the fingers of the other hand and then moved the fingers below the other hand for each hand. The combinations were:
 - a. Both index fingers.
 - b. Both index fingers and thumbs.
 - c. One hand with index finger and the other hand combined with thumb.
 - d. The same as in task 4.c. with thumb and middle finger.
 - e. All fingers with thumb for both hands.
5. They performed the same task as in task 4. while the second hand was from a different user on the left and afterwards on the right side.

3.4.5.3 Results

► Table 3.3 shows the recognition rates for all participants with mean deviations, for the tallest (height 1.98 m) participant, the smallest (height 1.52 m) participant, and the participants’ best and worst recognition rates. We annotated 84001 frames with 335561 touch contacts resulting in an average of 3.99 touch contacts per frame.

GEOMETRY ASSUMPTIONS. Although most of the participants had no experience with multi-touch interaction, they touched the surface in such a way that the finger orientation could be recognized correctly. For that reason the recognition rate was over 92% in the worst case for the geometry (1., 2.) and distance variance (3.) tasks. The average recognition rates of about 96% (task 3.) / 97% (task 1., 2.) confirm our geometry assumptions and the efficacy of the formulated constraints. For these tasks, false classification occurred mostly in arrangements with thumb in combination with ring finger or little finger, where the angle between the fingers was around 180°. Those arrangements occur when fingers were oriented quasi anti-parallel as illustrated in ► Figure 3.36. This observation sug-

Results confirm geometry assumptions of the constraints. False classifications were mainly caused by erroneous finger orientations or combinations with thumbs.

Participant	Geometry (1, 2)	Variance (3)	Conflicts (4, 5)
Average overall	97.5%, $\sigma = 0.48$	96.82%, $\sigma = 0.79$	90.55%, $\sigma = 1.05$
Tallest	98.1%	98.5%	92.3%
Smallest	95.0%	92.0%	90.1%
Best	99.5%	99.6%	95.4%
Worst	94.9%	92.0%	84.2%

Table 3.3: Recognition rates for all participants, the tallest participant, the smallest participant, and the participants' best and the worst recognition rates.

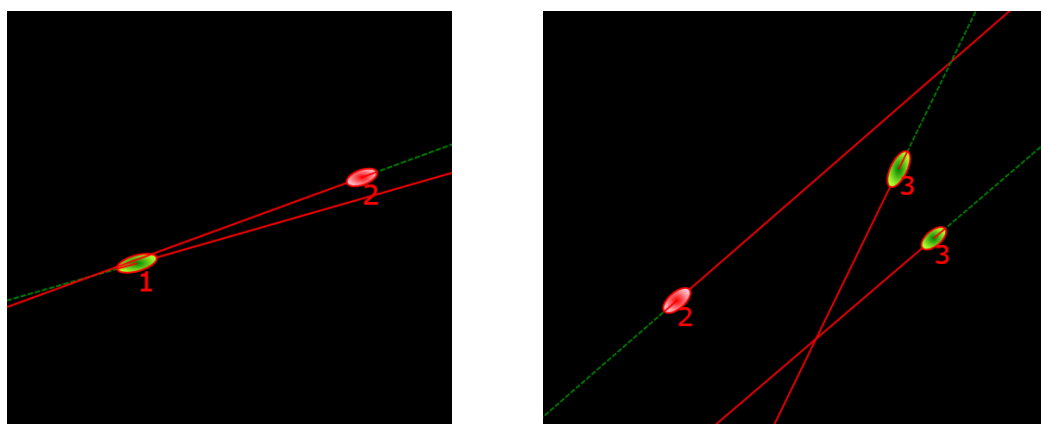


Figure 3.36: False classification with thumb and ring-finger / little-finger. An ellipse depicts a touch contact; the lines indicate the orientation and the numbers denote the recognized handIds.

gests extending the approach for handling the thumb separately, since it can take on constellations with fingers that the fingers alone cannot form. However, finger combinations with three and more touch contacts were recognized with high accuracy. This is due to the stabilization effect of adjacent fingers of one hand, where our adjacency distance includes up to three fingers. Also worth mentioning is that the recognition rates should be considered under the circumstance that the finger orientations were not detected with perfect accuracy through the tracking component. Certainly, this could be improved, but overall the detection was accurate enough to handle the wrongly detected finger orientations as little noise within the results.

CONFLICT SITUATIONS. Tasks 4. and 5. lasted between 30 seconds and 60 seconds each and the participants moved their fingers into a conflict situation at least twice, following the given combinations, for example, as illustrated in ► Figure 3.37. As expected, the recognition rates for conflict situations were worse than for tasks 1. - 3. but were still better than 84% in the worst case and about 95%

Recognition results for conflict case tasks were worse but still better than 84%.

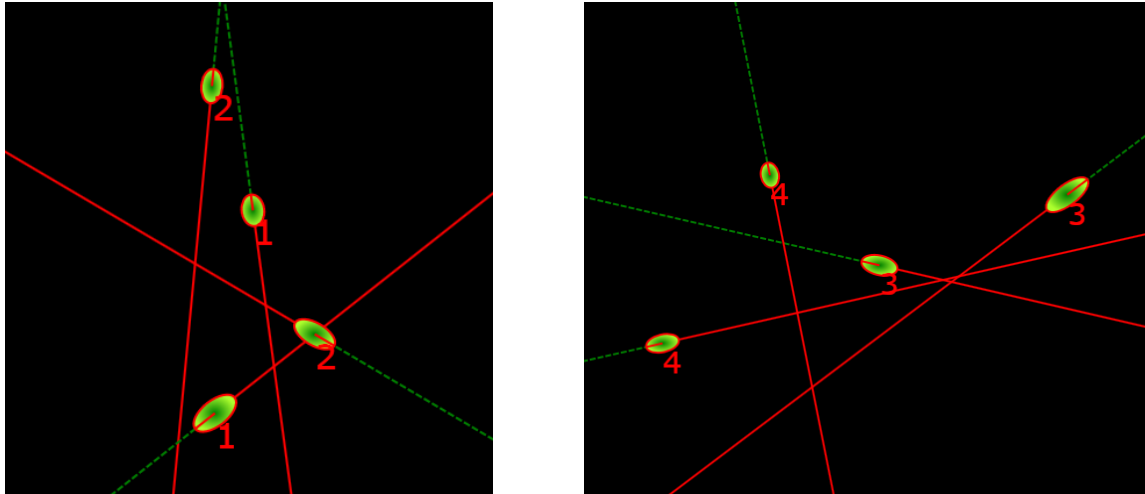


Figure 3.37: Examples of conflict cases where one hand is located beneath the other hand.

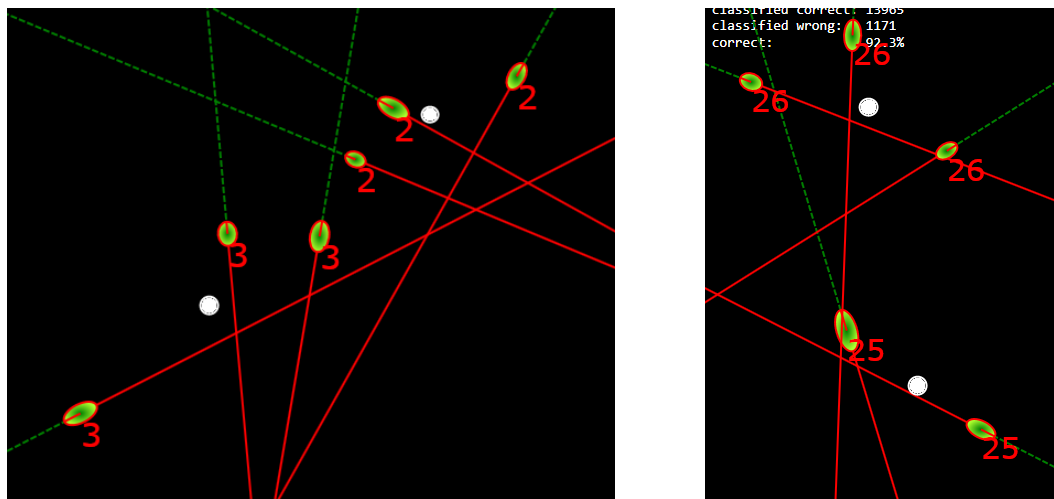


Figure 3.38: Each image consists of two hands with several fingers and their centroids. Centroids provide information to mark the fingers of two different hands.

in the best case. Here, the recognition process benefits greatly from the temporal adaptation and there is still potential for improving the temporal adaptation. If one hand moves below the other hand, then the mapping creates two finger clusters that move on the 2D-surface. We observed that the finger clusters sometimes merged to one big cluster, hence counting to a lot of falsely classified fingers. To cope with that issue, strategies could be applied that take the clusters over multiple frames into account. For example, two clusters can be assigned a confidence value for being distinct hands by virtue of the Euclidean distance between their cluster-centroids as sketched in ► Figure 3.38. Once distinct clusters have been identified, these confidence values, adhered to particular finger identifiers, then influence the decision whether to correct an assigned handId or not. Another source of falsely classified fingers was the recognition of finger orientation because users beware of getting in contact with the fingers of other users. Hence,

they sometimes moved their fingers to touch the surface under a perpendicular angle, which in turn led to wrong recognized finger orientations. In all, conflict cases like these are rather rare and are mostly unintentional. Therefore, the accuracy for these cases is not decisive for real world multi-user interaction.

Conflict cases as simulated in tasks are rare and mostly not intended.

3.4.6 Discussion

This section presented an approach to distinguish hands of finger touch contacts on a tabletop surface using only finger position in combination with finger orientation. Even though without perfect accuracy of the hand-mapping approach, the results show that the presented concept is sufficient to distinguish hands for two-handed input on interactive tabletops.

False recognition occurred mostly because of the thumb being more flexible as for the constellations with fingers. Therefore, enhancing the approach through special consideration of the thumb would improve the approach.

We also found that the temporal adaptation of our prototype was an important part and still bears a lot of potential for improvements, for example, through keeping statistics (e.g., confidence values) of more than one frame. Another approach is to identify fingers as being from different hands by means of the centroid of the identified hands.

3.5 Chapter Summary

The majority of applications and interaction techniques for digital tabletops only utilize x/y-coordinates from touch contact areas, which is a meaningful and reasonable design as this property is reliable and always available through any kind of touch sensor surfaces. This, however, obfuscates the view on the whole picture of finger input properties at users' side as well as designers and developers' side. Thus, much potential for interaction design is left unused, which could be otherwise employed to richly extend or realize novel and intuitive interaction techniques as we have shown in ► Section 3.2 and Sections 3.3.1 and 3.4.1.

To address this circumstance and improve the situation, this chapter has drawn the whole picture and established the first part of our design space for tabletop interaction beyond touch. This part includes interaction techniques that employ information explicitly expressed by users only through touching on a surface with their fingers. We first introduced and reasoned the two dimensions "finger input properties" and "touch sensor technologies" for the design subspace given in ► Table 3.1. Each combination in the design subspace indicates the

availability of a particular finger input property for a particular class of sensor technology.

Subsequently, we introduced and discussed the classes of sensor technologies with their functional principle, idiosyncrasies, advantages over other sensor classes, limitations, and implementation variants. In addition, for each class of sensor technology, we also discussed the given availability of each finger input property as well as limitations entailed by different sensor technologies. As part of this discussion, we contributed experiences and insights from our own exploration into the Pulsed Diffused Illumination technology and showed that it is a promising extension of the classic DI approach.

We then discussed each finger input property in ► Table 3.1 in terms of limitations, suitability, and insights from their usage for interaction techniques within the existing literature. As the properties finger orientation and hand distinction were under-researched, the remainder of the chapter presented our contributions for both finger input properties in order to complete the design subspace.

We discussed both finger input properties for interaction techniques and presented concrete interaction techniques based on these finger input properties. For both finger input properties, we contributed efficient detection algorithms together with evaluation approaches and results in order to show that both approaches achieve high precision.

Chapter 4

Multimodal Tabletop and Tabletop Tangible Interaction

THE second design subspace in tabletop interaction beyond touch is characterized by the use of input modes, auxiliary sensors (cf. ► Figure 4.1), physical artifacts (cf. ► Figure 4.2), or mobile devices in addition to touch input, such as employed in the large fields of tangible interaction and multimodal interaction. In human-computer interaction, the words "multimodal interaction" literally mean using multiple modalities to realize user interaction with an interactive system and the words "tangible interaction" literally mean using physical artifacts within user interaction. Tangible interaction is often considered as part of multimodal interaction because physical artifacts can be deemed as an interaction modality. Another notion of multimodal interaction envisions the utilization of interaction techniques, which address different sensory modalities at the same time [Wechsung, 2014]. Such descriptions for multimodal interaction seems clear at the first glance but the devil – the ambiguity – is in the details as we will see in the remainder.

Let's first take a look at the aims of multimodal and tangible interaction research before we scrutinize the details and relevance to tabletop interaction. In everyday interpersonal communication with an individual or a group of individuals, humans intuitively make use of multiple ways or channels (in combination) to communicate with each other. That is, the act of communication is not limited to speaking but also includes, for example, performing deictic, metaphoric or sign gestures, eye gaze, or body/arm/hand postures. Thus, human-human interaction is multimodal per se and humans are fairly skilled in this form of interaction as such belongs to everyday activities and are trained from childhood onwards. The same applies to tangible interaction as humans interact with physical artifacts – which are tangible or graspable – in the real world every day.

How is multimodal tabletop / tabletop tangible interaction characterized in brief?



Figure 4.1: An ECG sensor attached to a user's leg and a WiiMote controller in the user's pocket.

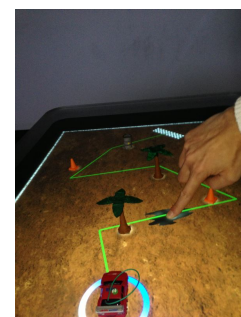


Figure 4.2: Physical artifacts on a tabletop surface for tabletop tangible interaction.

Humans developed highly skilled manual dexterity to grasp and use physical artifacts from childhood onwards. Multimodal and tangible interaction aim at enabling human-computer interactions that are closer towards such human natural behavior in order to exploit ways of interaction with which inexperienced, untrained, non-expert, or non-technical users are familiar with.

Focus on multimodal interaction literature related to tabletop interaction.

Due to the long history of over 30 years of research on multimodal systems and interactions by many research disciplines, it is not a surprise that many research works related to multimodal and tangible interaction (and systems) in general exist. In particular, there is much more literature on multimodal research that is not related to tabletop interaction than those related to tabletop interaction. This abundance and multiplicity of works make it challenging and laborious to sort out not relevant works and get a comprehensive overview of research results for the design of tabletop interactions beyond touch. This chapter addresses this circumstance with the aim to establish a focused overview over multimodal tabletop interaction / tabletop tangible interaction. Apart from that, the chapter's discussions address the following research questions with interaction design in mind.

- What is multimodal tabletop interaction / tabletop tangible interaction? How is it defined?
- How does it contribute to tabletop interaction beyond touch?
- What are the potentials for tabletop interaction beyond touch?

Tangible interaction is often considered as a modality of multimodal interaction.

As we have discussed before, tangible interaction can be deemed as part of multimodal interaction. However, within the field of multimodal tabletop interaction research, substantial attention has been given to research on tabletop tangible interaction wherein physical objects on the surface serve as an interaction modality in addition to touch input. This fact is mainly due to the feature of optical-based tabletop systems to track physical objects placed on the tabletop surface.

There is much more work on tabletop tangible interaction than for any other tabletop interaction modality. Therefore, we discuss tabletop tangible interaction and multimodal tabletop interaction in separate sections. Both sections first clarify the terms that are used throughout this dissertation by stating clear definitions. Afterwards, we provide comprehensible overviews and discuss both topics with the design of tabletop interactions beyond touch in mind.

4.1 Multimodal Tabletop Interaction

Contribution Statement: We contributed a prototype implementation of a multimodal tabletop interaction application called Surface-Poker to the field of multimodal tabletop interaction. The prototype made use of spatial on-surface gestures and addressed biosignals of users, which have not often been considered in the literature for multimodal tabletop interaction. The work was presented and published as a poster paper [Dang and André, 2010] at the ITS (Interactive Tabletops and Surfaces) 2010 conference.

<http://doi.acm.org/10.1145/1936652.1936701>

A broad range of different research domains has contributed to multimodal interaction in the past, for instance, human-computer interaction, signal processing, computer-vision, sociology, cognitive psychology, and many others. Due to so many different research domains and communities, the relevant terms such as modality, mode, medium, code, channel, or device received different meanings, which are subject to the particular research disciplines [Turk, 2014]. To avoid confusion caused by different terminologies and research / application domains, we state the following premises for this dissertation.

Many research disciplines have developed different meanings for multimodal interaction related terms.

1. We consider multimodal interaction from a human-computer interaction perspective.
2. Furthermore, we consider only multimodal interaction that includes a tabletop as the central part of the system, that is, multimodal tabletop interaction.

Premises to avoid confusion due to terms.

4.1.1 Terms and Terminology

Even within the human-computer interaction community, the meaning of relevant terms has been developed, refined, and used differently over time. Therefore, it is important to first establish an understanding of the relevant terms that

Ambiguous usage of Modality within HCI.

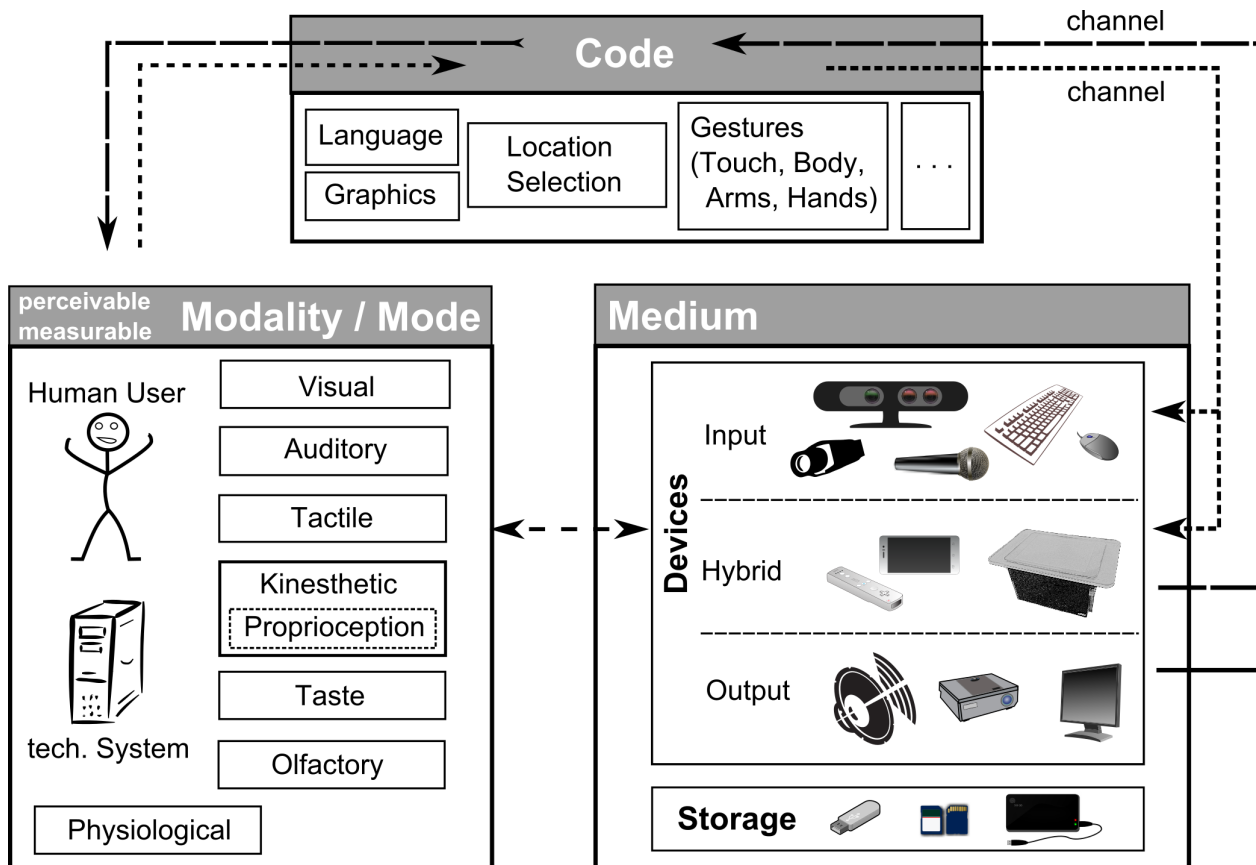


Figure 4.3: Overview of terms and components for multimodal interaction. A refined and updated scheme inspired by Maybury and Wahlster [1998].
(Source of cliparts: openclipart.org).

we use in the following. The most ambiguous usage pertains to the term "modality". Wechsung [2014] recently reviewed the literature and recapped three widely adopted usages of the term modality, that is, whether the meaning is rooted in physiological/human-centered perspectives, technology/system-centered perspectives, or both. Physiological perspectives relate modalities to human perceptual sensory while technological perspectives relate modalities to technical aspects such as information representation or interaction techniques.

This work adopts the human-centered as well as the system-centered perspective and disambiguates the usage of terms through the context of use as described later in ► Subsection 4.1.1.1. We take the understanding of Maybury and Wahlster [1998] of multimodal systems as a basis and refine the terminology to (1) include recent interaction devices and techniques and (2) to coalesce the denotations with their prevailing usage in tabletop research. ► Figure 4.3 sketches the terminology and their relations to each other that are used throughout this dissertation.

We adopt the human-
as well as
system-centered
perspective for the
meaning of terms.

MODALITY (PERCEIVABLE/MEASURABLE) / MODE. The term "perceivable / measurable Modality" or "Mode" can be either interpreted from a human user's perspective or a technical machine's perspective. From a technical machine's perspective, the term "measurable modalities" refers to physical signals or information that can be sensed or measured by a system through input devices or dedicated sensors. The dashed line between modality and medium in ► Figure 4.3 indicates that some of the modalities have a direct mapping to an input device. For example, the auditory modality can be sensed by means of a microphone; the visual modality can be sensed by means of cameras. However, kinesthetics and proprioception in ► Figure 4.3 are not (yet) directly available for technical systems as they relate to intrinsic human senses. The opposite applies to most physiological signals, which can be measured by appropriate sensors (e.g., EEG or ECG) and used for interactions but cannot be directly sensed by humans. Therefore, the measurable physiological modality is positioned indented in ► Figure 4.3. In the human user's perspective, "perceivable modalities" refer to receiving stimuli from human perceptual senses, such as visual, auditory, tactile, kinesthetic, proprioception, taste, or olfactory sense.

Modality from a technical perspective is measurable.

Technical modality can relate to input devices (Medium).

Modality from a human user's perspective is perceivable.

In comparison to the modalities listed in Maybury and Wahlster [1998, p.5], we separated the haptic sense into tactile, kinesthetic, and proprioceptive perception. The tactile perception, also called tactile sense, is related to stimuli received through nerve cells in the human skin, for example, when a finger has landed on a surface or when we perceive the friction of different surface materials through our fingers. The kinesthetic and proprioceptive sense relates to stimuli from skeletal muscles, tendons, and joints in order to provide a sense of physical movements. The proprioceptive sense is part of the kinesthetic sense and gives feedback about body pose and position in space and the relative positions of body parts to each other [Moen, 2006, p.12].

Tactile sense.

Kinesthetic and proprioceptive sense.

The separate consideration of tactile, kinesthetic, and proprioceptive sense help gain a better understanding of natural interaction styles such as multi-touch, bi-manual, full-body, or embodied interaction, as an increasing amount of recent research efforts have demonstrated. For example, Tan et al. [2002] provided kinesthetic cues on a touchscreen to yield 19% improvement in spatial memory. In a similar way, Kaufmann and Ahlström [2013] studied spatial memory and map-navigation performance on projector phones with consideration of kinesthetic cues. Jetter et al. [2012] conducted a study on panning UIs in which they observed a 29% increase in navigation performance in the touch input condition and explained this effect with kinesthetic and proprioceptive feedback. In terms of full-body interaction, one example out of many is given by Fogtmann et al. [2008] who investigated the bodily potential in interaction design on the theoretical foundation for kinesthetic interaction.

Considering tactile, kinesthetic, and proprioceptive sense help enable a better understanding of interaction related phenomena.

Two kinds of mediums: persistent storage and interaction devices.

MEDIUM. A Medium serves to carry, transmit, or transform information. ► Figure 4.3 distinguishes between two kinds of mediums. The first one merely serves as persistent storage for information, for example, flash drives or hard disks. This kind of medium is usually not an immediate part of multimodal interaction. However, the second kind of mediums is an integral part of systems for multimodal tabletop interaction. Similar to Maybury and Wahlster's [1998, p.5] distinction into output rendering and input processing, our scheme generalizes those entities as devices and differentiates between pure input (e.g., microphone, camera, or keyboard), pure output (e.g., display, loudspeaker, or projector), and hybrid devices (e.g., touchscreen).

Medium: input devices.

Input devices serve to sense or capture user input and to transform the input data into a form that is suitable to be processed by computers. These devices make use of sensors to continuously sample input signals and deliver sensor data (e.g., depth images, audio stream) or deliver event data (e.g., button press, mouse wheel spin). Hence, input devices can be figuratively seen as the "senses of machines for perception" of the modalities. Output devices serve to render information into the physical world intended to stimulate user's perception over the modalities. Hybrid devices provide input as well as output capabilities at the same time, for example, interactive tabletops, smartphones, or a Nintendo WiiMote controller.

Medium: output devices.

Medium: hybrid devices.

Moreover, the dashed line between modality and medium in ► Figure 4.3 indicates that some of the input and output devices have a direct mapping to one modality. For example, a microphone is usually employed to sample user's voice, thus receive auditory input from a user, whereas the loudspeaker is used for the opposite direction to stimulate the auditory modality of the user.

Code, Channel vs. Medium and Modality.

CODE / CHANNEL. The last components in ► Figure 4.3 are "Channels" and "Codes" (also called "Encodings" [Blattner and Glinert, 1996]). A code functions as an adapter that combines modality and medium through a certain representation of information. For example, information may be represented as written text, spoken words, braille characters, or even a sequence of lip and mouth movements.

Pathways represent connections between the entities.

A (communication) channel represents a "*particular pathway*" [Turk, 2014] from a medium through a code to a modality (or the other way round) through which information is transmitted. For example, if a text is rendered on a display to be read by users, then the pathway of information may start at the display and end at the visual modality. Another pathway could start at the visual or kinesthetic modality and end at the (depth) camera device if users enter text to the system by drawing letters in the air with a finger.

4.1.1.1 System-Centered Modality

According to Wechsung [2014], there is not only one system-centered understanding of modality in terms of a commonly agreed definition. The meaning of modality is rather subject to particular system-oriented views on multimodal systems. Hence, in order to provide further insights into the meaning of modality in the research literature, we carried out a review of literature related to multimodal tabletop interaction with a system-oriented understanding of modality. The results show that researchers often use modality as an "umbrella term" or as an attribute for other multimodal interaction related terms and aspects of ► Figure 4.3.

Usage of system-centered modality in the multimodal tabletop interaction domain.

In the following, we provide a compilation of relevant citations (from the literature review) to exemplarily illustrate the prevailing usage of the most often occurred interpretation either as input/output modality or interaction modality.

Input or output modality referring to particular devices, mediums, or channels:

- *"When a tabletop only supports direct-touch as an input modality ..."* Banerjee et al. [2011]
- *"While a direct-touch input modality may ..."* Forlines et al. [2007]
- *"... the indirect input modality of the circular disks."* Haller et al. [2010]
- *"Which pen, touch, or pen+touch gestures should a system support, and why? ..."* Hinckley et al. [2010]
- *"... how to use new modalities ... if each input modality offers ..."* Hinckley et al. [2010]
- *"... speech appears to be an ideal input modality ..."* Tse et al. [2006]
- *"... two-level tapping as a reliable input modality. ..."* Pedersen and Hornbæk [2014]
- *"Touch-based interaction is the oldest and still most used modality."* Ardito et al. [2015]

Interaction modality referring to particular communication channels or codes:

- *"Physical user action as an interaction modality ..."* Haller et al. [2010]
- *"... we define a richer touch interaction modality ..."* Murugappan et al. [2012]

Murugappan et al. [2012]	● "... new applications of this new touch modality."
Nowacka et al. [2013]	● "... afford the use of movement as an expressive output modality."
Arif and Stuerzlinger [2013]	● "... pressure-sensitive touchscreens as an alternative interaction modality ... pressure as a modality ..."
Pedersen and Hornbæk [2014]	● "Touch has become a prominent modality for interacting ..."
Sharma et al. [2011]	● "... using combined gesture and speech for the graphics manipulation, instead of either modality alone. ..."
Bellucci et al. [2014]	● "... interact with more than one finger at time, taking into account hand movements as well as gestures. This modality ..."
Bianchi-Berthouze et al. [2007]	● "... facial expression is the most powerful modality ..."
Bianchi-Berthouze et al. [2007]	● "... thus showing posture as a very powerful communicative modality."
Fogtmann et al. [2008]	● "... free and expressive full-body movement as an interaction modality ..."

Usage of modality in examples as an umbrella term, and to serve as a shortcut for medium, code or channel.

All the given citations make sense in their particular context. They use the term modality as an easy to understand name or shortcut for medium, code or channel. For example, the words "*speech input modality*" [Tse et al., 2006] in the context of a system that accepts speech commands as well as touch input creates a clear understanding of the available communication channels. Here, the usage of modality actually describes the channel that starts at the microphone (input device), goes through the language code, and ends at the machine's (measurable) auditory modality. Often, the term modality is used to address different codes (or languages) as, for example, in [Bianchi-Berthouze et al., 2007], whose intention was to emphasize interaction through facial expressions or postures.

Disambiguation of modality through perceivable / measurable modalities for human-centered meaning, otherwise system-centered usage.

DISAMBIGUATION OF MODALITY. We carry on with the system-centered usage of the term modality because it is common and prevailing within the tabletop research community. While the usage of modality as an umbrella term or within a shortcut is reasonable in the particular context, it is, however, a source of ambiguity for assessing a clear definition with the historical human-centered understanding of modality in mind. Therefore, we disambiguate the usage by means of the term "perceivable / measurable modality" when we refer to the human-centered meaning of modality. Thereby, the meaning and related perspective become clear, that is, whether modality refers to a perceivable / measurable modality or whether modality refers to the system-centered usage of the term.

4.1.2 Definition of Multimodal Tabletop Interaction

The research literature provides different views and definitions for multimodal interactions or systems that enable multimodal interactions, which depend on the perspective's root (human-centered or system-centered) as well as the certain usage of the term modality [Wechsung, 2014].

To provide a consistent understanding used throughout this dissertation, we give definitions based on ► Figure 4.3 and the previously discussed terminology. Similar to the interpretation of ► Figure 4.3, the following definitions are inspired by the understanding of Maybury and Wahlster [1998, p.5] and are insofar sufficient as to discuss multimodal interaction within the scope of this dissertation. The following definitions are consecutively constructed and correlated, that is, we first define multimodal systems and interactions in general and give, on this basis, a definition for multimodal tabletop systems and interactions.

Definition 4. *Multimodal interaction and according systems refer to interactions and systems that enable users to input and interact through multiple perceivable modalities, devices or multiple channels independently, tightly coupled, simultaneously, or serially. Furthermore, multimodal interaction goes beyond sole traditional GUI interaction with mice and keyboards.*

PRECLUSION. The last prerequisite in ► Definition 4 is by reason of multimodal research's history in which the standard interaction devices were keyboards and mice combined with a graphical user interface workstation. Without excluding those devices, virtually all conventional computers or laptops have to be considered as multimodal systems, which in turn renders the definition unnecessary. The latter excludes only the combination of keyboard and mouse. However, interaction through, for example, keyboard and speech, or keyboard with mouse and eye gaze belongs to multimodal interaction.

INPUT / OUTPUT. ► Definition 4 clearly emphasizes interaction, which requires at least two different modalities for user input. In particular, systems that provide only one input channel and exploit multiple perceivable modalities for system output do not represent multimodal interaction systems but represent multimedia systems. For example, let's consider an instrumented room that provides a touch-display and a large projected display for visualizations together with music as well as sound and spoken language through loudspeakers. In addition, the room provides multiple different smells and ambient light colors, which can be controlled by software applications of the instrumented room. Here, input is

► Definition 4 excludes sole mice/keyboard GUI workstations.

► Definition 4 emphasizes interaction and user input.

Multimodal output.

solely enabled through touch input on the touchscreen. Such a system certainly provides multimodal output but does not enable multimodal interaction according to ► Definition 4 as only touch input is supported. If, however, the system would enable speech input as well, then the system enables multimodal interaction. For the sake of completeness, multimodal input alone is not sufficient for multimodal interaction as interaction requires output in response to the input.

Multimodal input.

Multimodal interaction.

Definition 5. *Multimodal tabletop interaction and according systems refer to interactions and systems according to ► Definition 4, which have a digital tabletop surface as central part and go beyond sole touch interaction.*

Since touch interaction on traditional tabletops is comparable with keyboard and mouse interaction for traditional workstations, ► Definition 5 consequently treats the standard input method the same way as for multimodal interaction in ► Definition 4, that is, keyboard, mouse, and touch input are regarded as standard input methods. For example, a user who operates a tabletop with only uni-manual or bi-manual touch gestures would (intuitively) not be regarded as interacting through multiple measurable modalities as touch interaction is the prominent basic interaction technique afforded by such systems. However, a user who speaks with the tabletop application or uses eye gaze to "point" at the tabletop's surface in addition to using touch gestures is clearly interacting through multiple (measurable) modalities with the tabletop application.

► Definition 5

excludes sole tabletop touch interaction.

4.1.3 Usage of Multimodal Tabletop Interaction

In the following, we assess the most often and most promising usages of multimodal tabletop interaction together with according research results. By this way, this section gives designers and developers a quick and helpful resource for design decisions related to multimodal tabletop interactions.

Coverage of research in this section.

LIMITATION AND CLARIFICATION. The research works selected for this section address systems according to ► Definition 5. As guidance for the selection, we preferred works that include at least two different input modalities in addition to sole touch input, as such works are more likely to address issues related to multimodal input.

We also do not discuss every interaction modality that would be possible with digital tabletops (or have not been investigated with digital tabletops) because such an extensive discussion would go far beyond the scope of this dissertation.

Moreover, many interaction modalities that became possible through recent developments are still subject to further research efforts. Finally, we concentrate the discussion on input channels but also examine output channels if they are particularly relevant for tabletop interaction.

COMPARISON OF MULTIMODAL OUTPUT. Before we start with multimodal input, there is one mentionable research work related to multimodal output that designers and developers should be aware of. McAdam and Brewster [2011] compared different types of feedback on tabletop surfaces, that is, none, visual, audio, and tactile (through mobile phones). They found that participants preferred audio or tactile feedback over visual feedback. Furthermore, by means of a target acquisition task, they found that the performance for audio and tactile feedback was comparable or better than in the visual feedback condition. The worst performance was measured for the condition with no feedback at all. Thus, using audio or tactile feedback as an alternative or in addition to visual feedback is viable and should be considered for the design of multimodal tabletop interactions.

Audio or tactile feedback can be viable alternatives to visual feedback.

INVESTIGATED INTERACTION CHANNELS. ► Table 4.1 gives an overview of the investigated interaction channels together with modality, medium, and codes according to the meaning of the terms in ► Figure 4.3. The third column (medium) also denotes whether a medium enables input, output, or both. Note that there are more research works and interaction channels available in the existing literature, however, without being investigated for tabletop interaction.

4.1.3.1 Speech

The most often researched channel in multimodal interaction (in general) is commonly called "speech". In turn, the large body of works dedicated to this channel shows the importance of speech for multimodal research [Blattner and Glinert, 1996, André and Martin, 2014]. Early efforts already came up with, for example, "Put that there" by Bolt [1980] or Quickset by Cohen et al. [1997]. In multimodal tabletop interaction, speech input can be employed in different ways. A common usage of speech is represented by spoken commands to extend touch interaction for tabletop applications. Spoken commands consist of keywords (e.g., "delete") or short sentences (e.g., "label as my objects"), which can be directly mapped to application commands or a sequence of commands (e.g., [Tse et al., 2007]). Together with sole touch input or touch gestures, spoken commands allow for the creation of efficient and powerful joint combinations of tasks, for example, using touch to make or narrow down an object selection and spoken commands to choose the operation that shall be performed on the selection. Spoken commands can also be used to complement in-air gestures above tabletops as shown

Speech as explicit commands.

	Interaction	Modality	Medium	Code – Applications
1	Speech	Auditory	Microphone (I), Loudspeaker (O)	Spoken command, Verbal utterance, Sentence
2	Pen	Visual, Tactile	Digital pen (I)	Text, Graphics, Gestures – User mapping, Pick-and-Drop
3	In-the-Air / Above surface gestures, pointing	Visual, Tactile, Kinesthetic, Proprioception	Camera (I), Motion capture system (I)	Pose / Gestures with head, body, hand, Arm, finger; Pointing / Location – User mapping, Pick-and-Drop
4	Bodily, Embodied	See row 3	See row 3	See row 3 + row 5
5	Bi-manual	Kinesthetic, Proprioception	Sensor surface (I)	Spatial memory, Spatial cues
6	Eye gaze	Visual	Eye tracker (I), Video (I)	Pointing/Location
7	Haptic	Tactile	Mobile device (I/O), Physical artifacts (I/O)	Pressure, Force, Malleability, Friction
8	Tangible Object	See row 3	Physical artifacts (I/O)	Location, Pressure, Force
9	Biosignal	Physiological	Heart rate sensor / ECG (I), EEG (I), EMG (I), EDA/GSR (I), Respiration rate (I)	Gestures – Supportive / Extend other interaction channels

Table 4.1: Classification of interaction channels for multimodal tabletop interaction including modality, medium, and code. (I) = Input, (O) = Output, (I/O) = Input + Output.

in [Mehlmann et al., 2014], cf. ► Figure 4.4. In addition to explicitly articulating commands to applications, spoken commands serve as implicit communication to collaborators as found in [Tse et al., 2007], which should be considered in interaction design. Furthermore, the choice of suitable microphones, their installation as well as the capability and the quality of speech recognition must be carefully considered in the design. Otherwise, the interaction would suffer from recognition of false positives as reported in [Sharma et al., 2011]. Speech can also be employed in an implicit and unintrusive manner in which verbal utterances of a group are recorded and analyzed in real-time in order to adapt applications accordingly. For example, Martínez et al. [2011] employed a 7-channel radial microphone array to determine from which direction audio input was originated. They further captured audio input and determined the speech content as well as verbal participation in order to adapt their collaborative learning envi-

Speech in an
unintrusive manner.



Figure 4.4: Multimodal game input using speech and in-air gestures.

ronment called "Collaid", for instance, to provide hints or help in a knowledge-linking task if speech input has identified particular learners who need assistance. A similar aim was pursued by Bachour et al. [2010] who captured verbal participation by means of three microphones in order to balance group collaboration, for instance, to indicate whether a participant is over-engaging in a conversation.

4.1.3.2 Pen

Digital pens have two significant properties that make them well suited to complement touch input for tabletop interaction. First, they enable input on surfaces at pixel-precision in contrast to the large contact area of a finger touch contact. Second, they provide affordances that most humans have learned to recognize beginning with the first day of school. Humans further develop and master their skills to use pens over the years.

Pen as versatile complement to touch input.

Hence, tabletop interaction with a pen was the second most researched input channel, for example, to input small portions of texts or graphics. Haller et al. [2006] employed an Anoto¹ pen together with an augmented tabletop environment to enable drawings on digital as well as physical paper in discussion and brainstorm meetings. Paths, which were drawn with the pen, were recorded in the system and either visually projected on the physical paper or rendered on the digital media on which the pen has drawn to. In addition to that, they provided physical tool palettes that can be interacted on with the pen, for example, to choose different colors or drawing tools. They also implemented a tool that makes use of the pen as a clipboard, which was first presented by Reki-

Pen interaction techniques.

¹ <http://www.anoto.com>

moto [1997] as the "Pick-and-Drop" metaphor. The digital pen plays the role of a portable clipboard that has the ability to carry a copy of digital information (by picking up with the pen) and to drop the information on another (physical) location.

In terms of pen tools, Hinckley et al. [2010] investigated the possibilities of digital pen tools as a complement to tabletop touch input. For this, they observed human behaviors with physical paper and notebooks in order to develop a prototype for note-taking and scrapbooking of materials. Based on their exploration of the prototype, they propose a division of labor between pen and touch for interaction design. Pen input should be used for writing and touch input should be used for manipulation tasks. Both together, that is, pen+touch, enable new bi-manual tools, for example, holding an object while using the pen as a knife to cut the object or to "peel off" a copy of the object. The efficiency and performance of such bi-manual tools are supported by an earlier experiment of Brandl et al. [2008] who conducted a comparative study with the conditions bi-manual touch, pen, and touch+pen. Their results suggested that bi-manual pen+touch input was *"superior in terms of speed, accuracy, and user preference"*.

Pen+touch =
bi-manual tools.

In comparison to touch input through human fingers, pen input causes less occlusion at the contact points and enable adaptation of the user interface with respect to occluded areas. For example, Brandl et al. [2009] and Doeweling et al. [2013] realized pen interactions, which exploit areas (depending on pen input properties) that are much more likely to be visible to users for displaying pie menus around the pen contact coordinates. As a side note, touch input would enable the same functionality if the finger orientations of touch contact areas would be supported by the touch processing software of the sensing system.

Pen cause less
occlusion.

Finally, digital pens that are wirelessly connected (e.g., through Bluetooth) to a tabletop system can be individually identified from each other. Doeweling et al. [2013] took advantage of this fact and investigated the use of digital pens (together with further user identification mechanisms) to realize user-based and role-based interactions within a situation analysis and planning application for disaster management.

User mapping through
pen ID.

4.1.3.3 Above Surface Gestures / Pointing

Tabletop interactions can also take place above tabletop surfaces, which open up a broad range of interaction channels. In order to enable such kind of interactions, appropriate input devices and recognition approaches are needed for capturing and analyzing the space above tabletop surfaces.

Input devices.

ENABLING TECHNOLOGIES. The simplest approach makes use of an infrared or color camera that is either mounted above the tabletop (e.g., [Agrawala et al., 1997, Buchmann et al., 2004, Spindler et al., 2009]) or mounted below the tabletop surface (e.g., [Hilliges et al., 2009]) directed towards the tabletop surface. Installations in which the camera is mounted below the tabletop surface are more complex and require special hardware components as demonstrated by Hilliges et al. [2009]. The position and pose of fingers, hands, arms, or heads of users are recognized by means of computer-vision approaches, which either exploit human skin color or the shape of moving parts within images together with connected component analysis on the captured images. However, accuracy and robustness of such approaches to determine 3D positions above the tabletop surface are often low due to, for example, limitations of computer-vision algorithms, camera noise, camera resolutions, or occlusions. More robust approaches, however, require users to be instrumented with small markers [Buchmann et al., 2004] or gloves [Agrawala et al., 1997].

Infrared or color camera.

The most robust and precise approach, however costly, employs professional motion capture systems with multiple cameras (e.g., Vicon²) such as proposed, for example, by Yao et al. [2006], Banerjee et al. [2011], Gjerlufsen et al. [2011], Wagner et al. [2013]. Motion capture technology provides sub-millimeter accuracy [Gjerlufsen et al., 2011] and tracks users or objects at a high rate (≥ 100 Hz) within a 3D volumetric space. However, this technology requires (1) users to be instrumented with passive retro-reflective infrared markers and (2) precise preparation / calibration of the system to each individual computing environment.

With the advent of low-priced consumer depth cameras, such as the Microsoft Kinect, the technological possibilities for research in terms of interaction above the tabletop surface have been fruitfully enriched. As a result, many research works emerged, which employed those cameras to investigate tabletop interaction above the surface (e.g., [Martínez et al., 2011, Martinez Maldonado et al., 2012, Murugappan et al., 2012, Haubner et al., 2013, Spindler et al., 2014]). The general setup of such systems has been discussed in ► Subsection 3.1.2.1 for detection of touch contacts in which the depth camera is usually mounted above the tabletop surface with a top-down view on the whole tabletop surface (cf. ► Figure 4.5). Since the camera placement is the same as for the aforementioned color/infrared camera approach, recognition on depth images also suffers from occlusion issues, for example, due to multiple body parts or objects located one upon the other.

In contrast to color/infrared images, depth images provide depth values through the pixels, which (1) enable easier and more robust recognition of body

² <http://www.vicon.com>

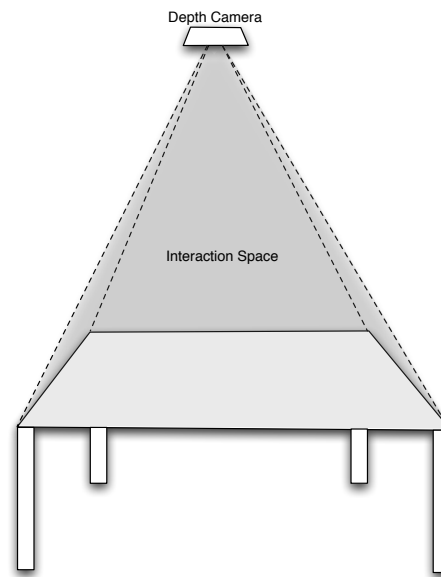


Figure 4.5: A depth camera setup for tabletop interaction above the surface.

Assessment of depth
camera approach.

parts and objects and (2) allow determining the distances between the tabletop surface and the detected body parts or objects as elaborated by Haubner et al. [2013]. The depth camera approach represents a trade-off between the color/infrared camera approach and the motion tracking approach in terms of recognition possibilities, accuracy of recognition, obtrusiveness for users, system complexity, and installation effort.

Noteworthy to mention, there is also a motion tracking approach that make use of inertial sensors (i.e., xsens³). Systems that employ this approach have not been used for multimodal tabletop interaction but may be a promising direction for research efforts.

INTERACTIONS. Interaction techniques above the tabletop surface can be distinguished between those that take the viewing perspective of users into account and those that do not. The majority of the investigated interaction techniques belongs to the latter and requires only the positioning of body parts or objects. By means of such information, finger and hand postures can be recognized such as demonstrated by Murugappan et al. [2012]. Finger and hand postures further enable recognition of in-the-air gestures as well as combinations of on-surface gestures with in-the-air gestures (e.g., Pick-and-Drop [Haubner et al., 2013]).

Height above surface
as DoF.

The height between fingers/hands and the tabletop surface offers an additional degree of freedom for interactions, for example, to adjust values or manipulate controls (e.g., zoom gesture) by means of the height as proposed by Haubner et al. [2013]. Furthermore, the height of objects above the surface in

³ <https://www.xsens.com/>

combination with its positioning (yaw, pitch, roll) in space enables meaningful adaptation or augmentation of the appearance of objects. For example, Spindler et al. [2009] proposed approaches for augmentation of physical paper through projection or appropriate visualization on tablet displays [Spindler et al., 2014] in order to provide intuitive methods for exploration of volumetric content above the tabletop surface. For this purpose, they modeled multiple content layers that were logically located at different heights above the tabletop surface. Depending on the height and positioning of displays, they rendered either different content or merged content layers/slices on the displays.

Augmentation of objects depending on positioning.

Multiple content layers in space above tabletop surface.

By continually observing body parts, tracking applications are able to model mappings of interactions (e.g., touch contacts or in-the-air gestures) to particular users to a certain extent [Martínez et al., 2011, Murugappan et al., 2012, Martinez-Maldonado et al., 2013]. Actually, most mapping approaches based on depth/color/infrared cameras employ a heuristic inference from recognized body parts to user positions at the tabletop sides. In particular, the mapping does not identify users but provides positions around the tabletop. In order to be reliably used for interaction, users have to keep their positions around the tabletop. Furthermore, body parts of different users should not approach each other too close as stated by Murugappan et al. [2012].

User mapping.

Motion tracking systems are superior in terms of user mapping as they can also track users if the users are appropriately instrumented. Banerjee et al. [2011] employed the advanced features of motion tracking systems to track users' heads in order to evaluate perspective aware in-the-air pointing at distant out-of-reach objects with the index finger.

Pointing at distant objects.

STEREOSCOPIC 3D. The volumetric interaction space above the tabletop facilitates interaction techniques through stereoscopic 3D vision such as investigated by Agrawala et al. [1997], Takeuchi and Sugimoto [2012], Ch'ng and Cooke [2015], or Bruder et al. [2013]. However, the requirements are considerably higher than in the hitherto discussed techniques. In order to render stereoscopic output for a user, the perspective from the user's head (i.e., her eyes) needs to be known to the system. This requires appropriate instrumentation as reasoned by Grossman and Wigdor [2007], for example, by means of trackable glasses that the users have to wear. Such glasses are required by most systems for realizing stereoscopic 3D output. In addition to that, the output or display technology of a tabletop surface has to support stereoscopic 3D output.

Stereoscopic interaction above the tabletop surface and its requirements.

Stereoscopic interaction above the tabletop surface is further limited to a few (mostly one) users because the stereoscopic 3D output has to be rendered for the particular viewing perspective of each user individually. This limitation negatively affects, for example, collaborative tasks and applications in which multiple users around the tabletop have to work on the same workspace.

Human perception
related effects.

Apart from technological requirements and limitations, researchers also identified human perception related effects that have to be considered in interaction design. In contrast to touching a physical 2D surface (of a tabletop), touching a virtual surface in 3D space does not provide any tactile feedback to the fingers, which "*leads to potential confusion and a significant number of overshoot errors*" [Colley et al., 2015]. For example, Chan et al. [2010] showed that direct-touch interactions for such 3D surfaces, also called intangible displays, "*performs poorly even for simple target acquisition tasks*". Their results revealed that users were missing an indication of when to stop moving the finger towards the surface or an object not only because of the lack of feedback but also due to technological issues. When users' fingers passed through the virtual surface, they could not see it as the fingers occluded the stereoscopic projection. This is because their eyes focused on the fingertip after passing through the surface, which destroyed the stereoscopic effect of the projection. This problem was also addressed by Valkov et al. [2011] who analyzed the relation between 3D positions of stereoscopic objects and its actual touch points on a surface. They found that users tend to touch a location that is between the projections for both eyes "*with an offset towards the projection for the dominant eye*".

4.1.3.4 Eye Gaze

Eye trackers require
calibration to users.

Research for gaze-based tabletop interactions is a young topic with only a few works, however, with promising results in terms of interaction. The interaction techniques from those research works should be considered with care as they merely address small tabletop surfaces. Detection of users' eye gazes is performed by means of specialized hardware, so-called eye trackers, which can be either mounted in front of users [Pfeuffer et al., 2015] or mounted on the users' heads (cf. ► Figure 4.6). All commercially available eye trackers have in common that they require a calibration procedure in which the eye tracker is adjusted to individual users before the recognition of eye gaze can be reliably conducted. This fact complicates the typical walk-up-and-use scenarios of tabletops. There are research works that address this issue for vertically mounted public displays. Vidal presented an approach called "Pursuits" in her Ph.D. dissertation [Vidal, 2014], which correlates the eye movements of users in front of a display against the trajectories of objects displayed on the display to avoid the calibration step. Nobody has investigated this technique for interactive tabletops but it represents a promising direction for further tabletop research.

In addition to the calibration procedure, the number of eye trackers limits the maximum number of users who can interact simultaneously on a tabletop workspace. However, advances in eye tracker technology and the availability of low-priced products will likely ameliorate both issues in the future.

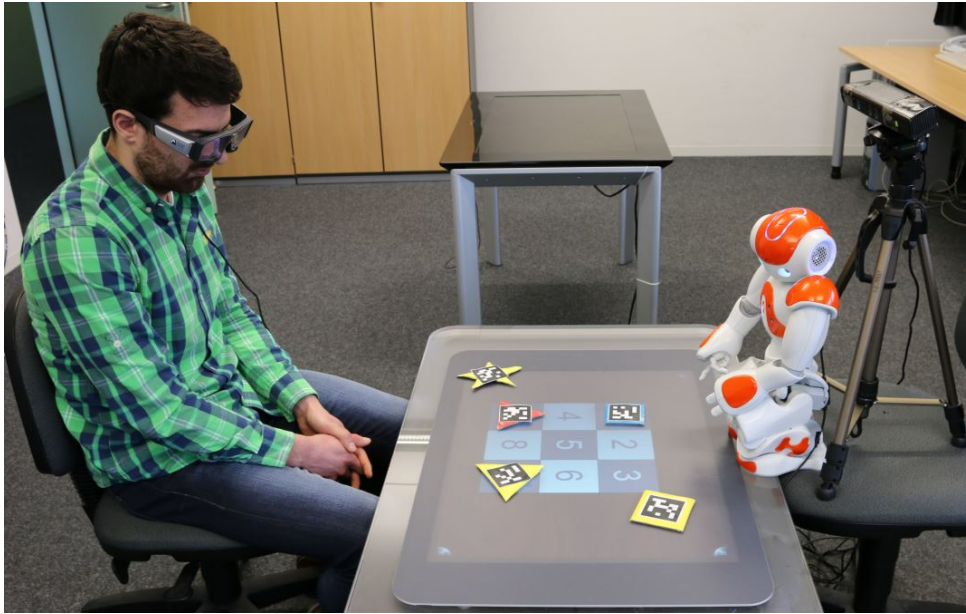


Figure 4.6: A user wearing a mobile eye tracker.

INTERACTIONS. Designing tabletop interaction solely on eye gaze without touch input comes at a (probably high) cost as also discussed in [Pfeuffer et al., 2015]. Such interactions would give away the intuitive direct-touch input (input/output are at the same location) in favor of an indirect input mechanism (input/output locations are different).

Hence, existing research works propose to design (eye) gaze-based tabletop interactions in a complementary manner, that is, eye gaze should be used to extend interaction techniques (e.g., touch and gaze) instead of being the sole interaction channel. For instance, Pfeuffer et al. [2015] and Voelker et al. [2015b] studied interaction techniques in which touch input was used for local manipulation of objects on the horizontal surface in front of the user. Eye gaze was used for selecting objects on remote surfaces [Pfeuffer et al., 2015] or switching between different surfaces [Voelker et al., 2015b]. If objects on remote surfaces were selected, then local touch manipulations can be performed on the selected objects on the remote surface.

Another application domain for gaze behavior is tabletop games as demonstrated by Yamamoto et al. [2011] who employed gaze behavior to make a digital card game more challenging. By observing users eye movements, their game character snatched cards on the game board before the user was actually able to reach it. Users, therefore, had to develop strategies to win under such circumstances. Gaze behavior can also be used for post-analysis of study results in order to gain a better understanding of user behavior during interaction sessions such as proposed by Ch'ng and Cooke [2015] and Piper and Hollan [2009].

Touch for local manipulations. Eye gaze for distant remote surfaces.

Influence tabletop games through gaze behavior.

Analysis of gaze behavior to gain an understanding of user behavior.

4.1.3.5 Haptic, Tangible Objects

Output modality.

As indicated in ► Table 4.1, haptic channels for multimodal interaction are commonly provided through the tactile output modality, which humans perceive via their tactile sense. The most prominent usage of the tactile channel is located in the tabletop tangible interaction domain, which we will extensively discuss in the next ► Section 4.2.

Tactile feedback through tangible objects.

Tactile feedback through motors in off-the-shelf mobile devices.

One example to briefly illustrate tactile output was presented by Marquardt et al. [2009] who simulated tactile feedback through a small building brick, similar to a puck, in combination with terrain map applications. Depending on the position of the puck on the map and the particular terrain information on the map, different types of tactile feedback, such as friction, height, texture, or malleability, were conducted to the user's index finger by means of a servomotor in the puck. Another mentionable approach to provide tactile feedback employs off-the-shelf consumer mobile devices, such as smartphones or game controllers (e.g., Nintendo WiiMote). For example, McAdam and Brewster [2011] realized tactile feedback through different vibration patterns generated by the vibration motor in a smartphone.

4.1.3.6 Kinesthetic / Proprioception

Brief refresh of the terms kinesthetics and proprioception.

Researchers regularly studied kinesthetic and proprioceptive perception in human-computer interaction, for example, in [Balakrishnan and Hinckley, 1999, Tan et al., 2002, Begel et al., 2004], with increasing interest in the tabletop interaction community within recent years, for example, by Wu et al. [2011], Jetter et al. [2012], Klinkhammer et al. [2013], or Rädle et al. [2013]. As discussed in ► Subsection 4.1.1, the kinesthetic sense gives humans an awareness of the physical movements of their body parts while the proprioceptive sense enables humans to register their body pose and positioning in space as well as the mutual relative positions of body parts [Moen, 2006, p.12].

Guiard's Kinematic Chain Model.

Early work by Guiard [1987] – though not in the context of HCI but cognitive psychology – considered kinesthetics in bi-manual (i.e., two-handed) actions to propose a theoretical framework, which is known as the Guiard's Kinematic Chain Model (KCM). In this model, the left hand and the right hand play different roles in bi-manual tasks. The division of labor between hands depends on their particular roles. Moreover, Guiard's model considers motion to be created through several motors, which are arranged in series, in the left and right hand, hence, resembling a kinematic chain. The motors in such a kinematic chain communicate and cooperate with each other during the creation of motions.

A simple example from Guiard's work draws on human writing skills. When writing on a sheet of paper with a pen, the left hand – we consider a right-handed user – re-positions the paper repeatedly and at the same time serves as a reference frame for the right hand, which does the writing with the pen. Motions of both hands are perfectly coordinated and synchronized with each other during the whole writing task. Being aware of such reference frames and being able to coordinate and control movements in tandem with those reference frames require humans to employ kinesthetic and proprioceptive senses.

Series of motions in bi-manual writing explained through the KCM.

Later, HCI researchers picked up Guiard's KCM and investigated the model's validity in the context of HCI. For instance, Balakrishnan and Hinckley [1999] conducted a study with a Wacom⁴ tablet and two physical pucks to compare the performance of kinesthetic and visual reference frames in bi-manual input. Their results suggest that Guiard's reference principle holds for kinesthetic and for visual references. Hence, appropriate visual feedback help overcome missing or limited kinesthetic references.

In 1987 and the decade after, when Guiard conceived the KCM, mouse and keyboard were the predominant interaction devices. In contrast, touchscreens (as are widespread today) and tabletop surfaces are quite different in the character and amount of (bi-manual) hand movements for input. Furthermore, movements in space through above-the-surface interactions and bodily / embodied input around or at the tabletop increase the importance of kinesthetic references. Consequently, considering human kinesthetic and proprioceptive senses as a perceivable modality in interaction design promise for better performance and more natural interaction techniques.

Importance of kinesthetics and proprioception for today's and future interaction.

INTERACTIONS. As kinesthetic and proprioceptive senses are complex systems inherent in human beings, directly measuring them based on physiological signals and utilizing such data as input for interaction is challenging and not common in HCI. Although motion and pose of human body parts can be tracked through motion tracking systems, researchers explored and exploited kinesthetics and proprioception rather in the design of interaction techniques and corresponding user interfaces.

A prominent subject of research addresses the so-called "spatial memory" and how it improves task performance through appropriate interaction techniques and interfaces, for example, by Leifert [2011], Jetter et al. [2012], Rädle et al. [2013], Klinkhammer et al. [2013]. The term spatial memory and its reasoning originate from the large research field embodied cognition and spatial cognition of the cognitive sciences discipline [Barsalou, 2008, Goldinger et al., 2016]. Spatial memory involves cognitive processes that humans use to encode and remember the locations of objects in the environment (i.e., as mental representa-

What is spatial memory?

⁴ <http://www.wacom.com>

tions) [van Asselen, 2005] as well as the environment itself through cognitive maps [Darken P. and Peterson, 2014]. There are dedicated areas and interrelated systems in the human brain for spatial memory processes according to a large collection of research works, which led to a Nobel Prize 2014 awarded to O'Keefe, Moser, and Moser [Gibson and Mair, 2016]. Hence, the existence and the functionality of spatial memory have been commonly accepted in the research community.

Spatial memory
together with
kinesthetics /
proprioception.

Considering spatial memory together with kinesthetics / proprioception – in the design of interaction and user interfaces – is grounded on the premise that sensorimotor information from kinesthetic and proprioceptive senses reinforce cognitive processes related to spatial memory [Tan et al., 2002, Jetter et al., 2012, Rädle et al., 2013]. According to van Asselen [2005], such sensorimotor information belong to the type of spatial information that *"can be used to create a cognitive spatial representation of the environment and keep it in memory"*.

Movements serve for
kinesthetic cues.

An early study, which was conducted by Tan et al. [2002], demonstrated a 19% performance increase in the touch input condition in comparison to mouse input. Participants had to recall objects and their locations that they manually positioned on a touch-enabled display before the recall task. The authors concluded that the touch condition provided more "kinesthetic cues" to users due to direct manipulation through touch input.

Later, Jetter et al. [2012] extended the methodology to include navigation performance as well. They compared touch input with mouse input for so-called panning UIs and the combination of panning and zooming UIs on a Microsoft Surface tabletop. Similar to the results in [Tan et al., 2002], the touch condition showed 37% better performance of spatial memory and 29% increase in navigation performance for panning UIs. However, panning+zooming UIs did not offer improvements in the touch condition.

Rädle et al. [2013] also studied spatial memory and navigation performance through touch input, however, in comparison to egocentric body movements for zoom-able UIs. The egocentric body condition achieved improvements in terms of path length (47%) and task time (34%) but no improvements in spatial memory. Only after a recall experiment that followed up the first experiment has shown 27% improvement in spatial memory in the egocentric body movement condition. However, the comparability of the study results with those in [Jetter et al., 2012] must be taken with care and is at least limited, as the input size of a tablet device is much smaller than the large input space of the Microsoft Surface tabletop, thus, severely reducing the overall amount and accent of movements (and kinesthetic / proprioceptive stimuli).

A third study in this direction was conducted by Klinkhammer et al. [2013] who compared the spatial memory and navigation performance of panning UIs between touch input and body panning using a large tabletop surface. The workspace on the tabletop could be panned either through pen gestures with touch input or changing user's position ("Body Panning") around the tabletop. Their results show statistically significant improvements in spatial memory in the body panning condition but no difference in the navigation performance.

Overall, all three studies strongly support the hypothesis that the amount, quality, and accent of kinesthetic and proprioceptive stimuli have an effect on spatial memory. Hence, interaction techniques should be designed with kinesthetic and proprioceptive sense in mind if designated applications would benefit from better spatial memory performance.

4.1.3.7 Biosignals

The last category of interaction channels employs physiological data – also called biosignals – of human users, such as heart rate through ECG (Electrocardiography), respiration rate, EMG (Electromyography), EEG (Electroencephalography), or EDA / GSR (Electrodermal Activity / Galvanic Skin Response). Most of such biosignals are usually not consciously controlled by humans but change depending on bodily (re-) actions (e.g., heart muscle) or physiological and emotional states of a user (e.g., brain activity or skin conductance). In turn, this property of biosignals makes them well suited as an additional channel for supporting the other interaction channels in ► Table 4.1.

Physiological data are not consciously controlled.

While many research works related to the usage of physiological data in HCI are available, only a few can be found that employ biosignals for tabletop interaction. Certainly, the requirements and handling of sensors for data acquisition are one of the reasons. Data acquisition requires users to be instrumented with wired sensors (and their wires) and the sensors often have to be calibrated to individual users. Depending on the kind of sensors, such prerequisites require laborious work and time. For example, EEG and ECG require careful preparation of human skin/head for sufficient conductance.

Laborious instrumentation is required.

INTERACTIONS. Most of the existing tabletop research works employed biosignals to adapt application states in response to human physiological reactions, for example, in [Cincuegrani et al., 2016, Dang and André, 2010]. Cincuegrani et al. [2016] included EEG and ECG to continuously adapt a musical application that ran on the ReacTable [Jordà et al., 2007]. Touch and gestural input together with tangible objects were used as default interaction channels. In addition, EEG signals were used to control the synthesis of sounds (sonification) while the heart rate influenced the tempo (BPM) of sound generation. They conducted an em-

Biosignals as an additional supportive channel.

pirical study with the musical application and showed that an additional implicit interaction channel through physiological data increased motivation and immersion of the tabletop application.

In [Dang and André, 2010], we included the heart rate of users to calculate a nervousness level in order to investigate new user experiences within a tabletop poker game. Good poker players usually try to hide their feelings and set up a so-called poker face, that is, a face, which does not expose any (true) emotional reactions. They make use of emotional reactions usually as part of a bluff. By displaying a visualization of the players' level of nervousness on the tabletop surface, we introduced a novel game element. Players were able to realize whether an opponent shows an increased level of nervousness. However, an increased level of nervousness may also be part of a bluff, for example, induced by thinking of stressful situations.

Instead of influencing application states, Benko et al. [2009] utilized the muscle activity of human arms and fingers (measured by EMG sensors) to extract complementary properties of fingers and finger postures during touch interaction. By analyzing the EMG signals, they were able to recognize different gestures (pinch, throw, flick) as well as pressure levels applied by fingers. By this means, they realized several interaction techniques that would not be able with sole touch input, such as "Pressure-sensitive painting" or "Finger-aware painting".

Similar to the work of Benko et al., Al-Megren et al. [2015] employed EMG signals measured on the user's arm but utilized the signals for post-analysis of user interactions instead of extending immediate user interaction. In particular, they investigated and compared arm fatigue when solving a puzzle using touch interaction on a horizontal tabletop and a large vertical display.

MODALITY. The majority of biosignals have no direct match with the traditional perceivable modalities as given in [Maybury and Wahlster, 1998] (i.e., Visual, Auditory, Tactile, Taste, Olfactory) but suggests for a measurable modality for interactive systems. Therefore, we introduced a unilateral modality called "physiological" that is mostly only measurable by technical systems. Only a few can be measured without technological means (e.g., heart rate, respiration rate). Even though some physiological signals can be perceived (and controlled) by humans, such as respiration rate, it is in most of the cases not meaningful for typical or prolonged interaction.

4.2 Tabletop Tangible Interaction

Tabletop tangible interaction is an interaction modality equal to other interaction modalities of multimodal tabletop interaction. However, the distribution of research efforts among all interaction modalities is unequal in the sense that there has been much more attention given to tabletop tangible interaction. Research related to tangible interaction and interfaces surfaced in the late '90s with early work of Fitzmaurice [1996], Fitzmaurice et al. [1995] or Ishii and Ullmer [1997]. At that time, Weiser proposed the vision of "Ubiquitous Computing" [Weiser, 1991] and research efforts towards "Augmented Reality" [Wellner et al., 1993] emerged, thereby giving inspiration to many tangible interaction researchers [Ishii and Ullmer, 1997].

There is much more research for tabletop tangible interaction than for other tabletop interaction modalities.

Influences from Ubiquitous Computing and Augmented Reality.

Over the last years, tabletop tangible interaction has attracted many researchers of various disciplines such as computer science, product or industrial design research, user interface or interaction design, or arts. Interest grew rapidly and has led to an own research community, which sought to find conceptual models and frameworks [Fitzmaurice et al., 1995, Ullmer and Ishii, 2000], explore certain application domains (e.g., urban planning [Underkoffler and Ishii, 1999], music generation [Jordà et al., 2007, Bischof et al., 2008], or tabletop games [Haller et al., 2010, Leitner et al., 2010, Dang and André, 2013]), or get a better understanding of natural affordances [Fitzmaurice et al., 1995, Terrenghi et al., 2007] among many other topics.

Many different research disciplines involved in tabletop tangible interaction.

In this dissertation, we consider tabletop tangible interaction from a computer science perspective, focusing on technological aspects to realize human-computer interaction. The remainder of this section briefly introduces into tabletop tangible interaction and presents a scheme to categorize tangibles.

4.2.1 TUI, Tangibles, Tangible Interaction

The term tabletop tangible interaction embraces interaction techniques for interfaces that are characterized by (1) the usage of physical tangible artifacts (e.g., ► Figure 4.7 and ► Figure 4.8) as an intermediate medium between users and the tabletop and (2) by exploiting tactile, kinesthetic, or proprioceptive modalities through physical properties such as appearance, shape, size, surface quality, or surface texture. The corresponding user interfaces are referred to as "tabletop TUIs"⁵ [Ullmer and Ishii, 2000, Ishii, 2008], "graspable interfaces" [Fitzmaurice et al., 1995], or TTIs⁶ [Bellucci et al., 2014, Pedersen and Hornbæk, 2011]. The

How is tabletop tangible interaction characterized?

TUI, TTI, graspable interfaces.

⁵ Tangible User Interface

⁶ Tabletop Tangible Interface



Figure 4.7: Examples of physical tangible artifacts used in student projects: (a) Real clams; (b) Commercial plastic figures of "Warmachines" as game pieces.

latter term TTI is not as widespread as the term TUI and solely confines TUIs to those that employ a tabletop. As this dissertation covers only the genre of TUIs that centers on tabletop computing, we use the more widespread term TUI instead of TTI for customariness' sake and as a shortcut for tabletop TUIs [Ishii, 2008, p.5].

TUI is most commonly used.

4.2.1.1 TUI

Tabletop tangible user interfaces consist of at least two physical parts: (1) an interactive tabletop system and (2) physical artifacts that are called tangibles. Interactive tabletop systems provide a large surface on which the system displays the visual presentation of applications. Furthermore, the surface senses and identifies physical artifacts on the surface. These features offer rich possibilities to visually augment the physical artifacts through adapting the visual presentation on the surface at the locations where the artifacts were detected.

TUIs = interactive tabletop + physical artifacts (tangibles).

Detection of tangibles makes use of visual markers that encode a numeric value and orientation

Detection and identification of physical artifacts are based on markers of various sizes, which are glued or printed on the bottom of tangibles, cf. ► Figure 4.9. Such markers are intended for optical-based tabletop systems and usually encode a numeric value with an orientation indicator that tabletop surfaces can sense in order to enable recognition of the position, orientation, and identifier of a marker and to identify the corresponding tangible object. By this way, optical-based interactive tabletops are able to constantly render the digital model of the application on the surface while keeping input/output space synchronized.

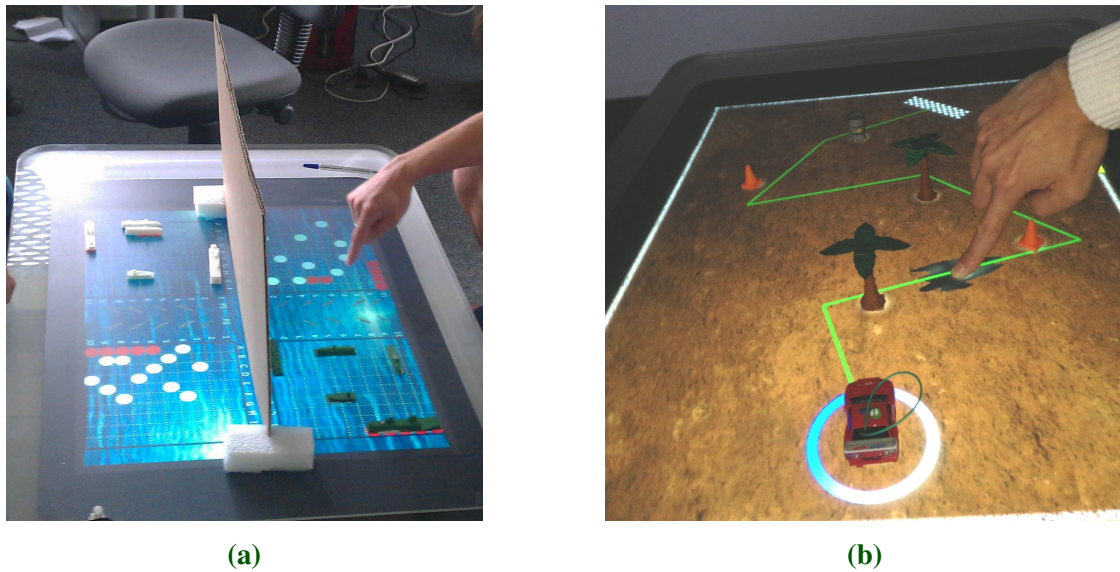


Figure 4.8: Examples of 3D printed physical tangible artifacts used in student projects:
 (a) Different colored and shaped tangible battleships;
 (b) Diverse miniature obstacles as tangible game pieces.

There are also approaches for capacitance-based tabletop systems to enable detection of tangibles. Such approaches make use of conductive material under the tangible artifact, which either connects the human skin of fingers to the tabletop surface (e.g., Disney Cars 2 AppMates⁷) or exploits a special marker pattern such as proposed by Voelker et al. [2015a]. The idea behind the approaches is to simulate touch contacts on the surface in order to map them to the tangibles.

4.2.1.2 Tangibles

Tangibles are physical graspable representations (cf. ► Figure 4.9) of digital objects and as such, they are tightly coupled or attached to the corresponding digital objects in state, function, or appearance. The physical form, size, or appearance of tangibles can be manifold, though they are typically designed or chosen to suit the intended application, task or metaphorical usage. For example, tangibles may take the form of abstract handles in order to be attached to arbitrary digital handles or objects as proposed in the concept for Bricks [Fitzmaurice et al., 1995]. In contrast to abstract and general-purpose forms, tangibles may also take the form of arbitrary everyday objects or specialized forms in coincidence with its digital counterparts such as Ishii and Ullmer's phicons (i.e., Physical Icons [Ishii and Ullmer, 1997]) or miniature obstacles in a tabletop game (e.g., TabletopCars [Dang and André, 2013]).

How are tangibles characterized?

⁷ <http://topbestappsforkids.com/best-kids-apps-disney-cars2>

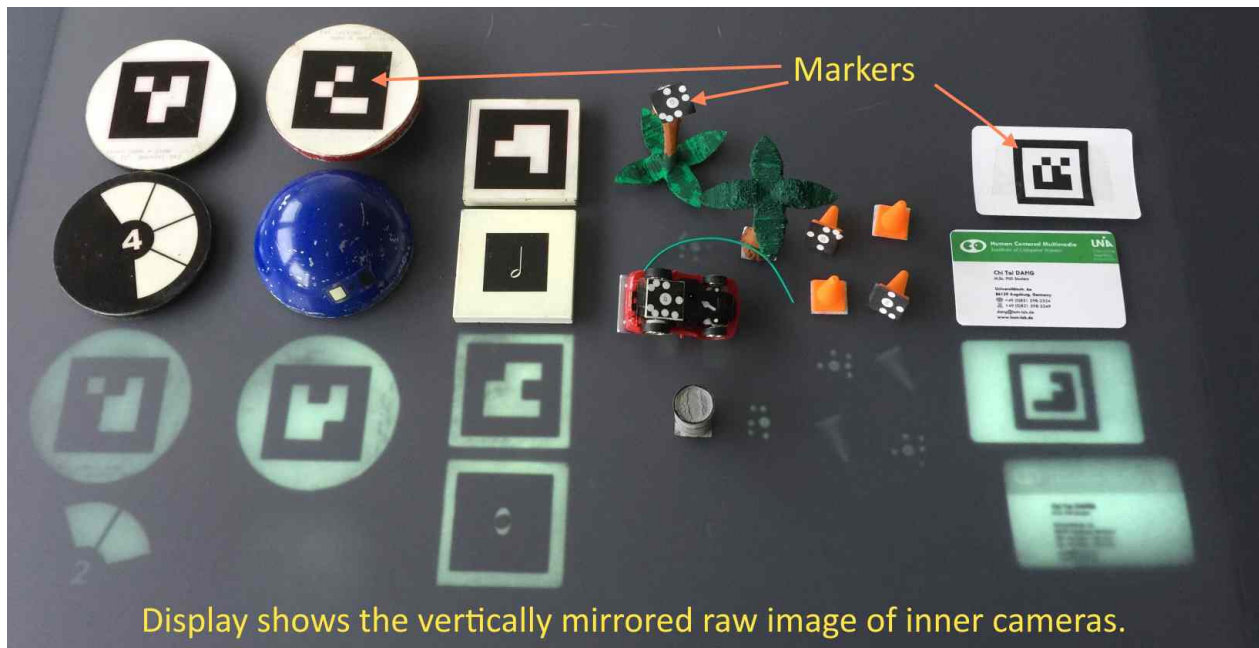


Figure 4.9: Different types of markers and tangibles. The upper part of the image shows tangibles and the bottom side of tangibles with markers. The lower part of the image shows the vertically mirrored raw image of the sensing surface.

Reusing of tangibles.

FORM AND FUNCTION. Tangible objects of abstract or generic forms have the advantage to be usable as a general-purpose tool/handle, which can be used in more than one TUI application. However, such forms come with significantly reduced quality of haptic and physical representation and thus lack of indication of functions. Ishii [2008] denoted such physical representation and the indication of functions as special-purpose-ness.

Application developers and interface/interaction designers have to carefully balance the appearance between general-purpose forms and special-purpose-ness. The ultimate goal is to find a trade-off that best suits user's needs and the addressed TUI application.

Balance between
general-purpose
forms and
special-purpose-ness.

Examples of good trade-offs are the SLAP-widgets proposed by Weiss et al. [2009], cf. ► Figure 4.10. SLAP-widgets are silicone based translucent tangibles. Some of them can be used as general-purpose handles for different digital objects while being specific enough in terms of form to clearly indicate the function. For example, the keypads (widget (a)) in ► Figure 4.10 with their buttons and the adjustable slider (widget (b)) in ► Figure 4.10 provide specialized functions, which aim at special-purpose-ness and are quickly understood by users. Both widgets may also be attached to a multitude of digital buttons or sliders, which create a trade-off to general-purpose use.

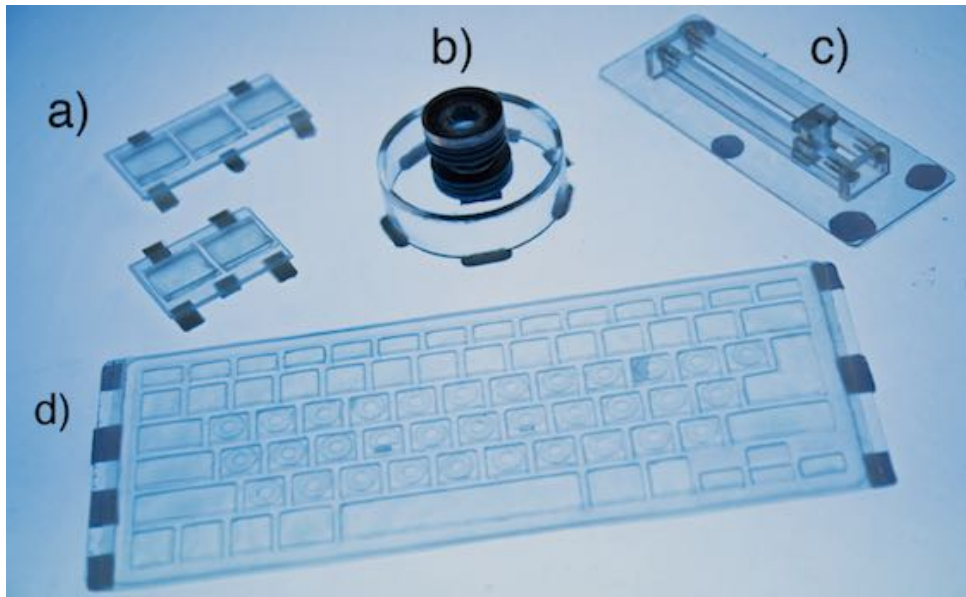


Figure 4.10:
Different types of
SLAP-widgets by
Weiss et al.
[2009].

FABRICATION OF TANGIBLES. Tangibles of abstract forms, such as bricks, can be fabricated easily, for example, realized as low-fidelity prototypes using paper, scissor, and glue. In contrast, miniature obstacles (► Figure 4.8b) or realistic play pieces (► Figure 4.7b) usually require off-the-shelf components, laborious handcraft work, or high fidelity prototyping approaches.

Low-fidelity
prototypes.

In particular, tangibles that constitute mechanical parts and electronic motors require fine-grained and high-precision components, which are quite difficult and laborious to handicraft. Here, industrial fabrication is usually cost-intensive if only a few tangibles have to be produced, which is typically the case for research prototypes. However, this issue has been greatly mitigated with the advent of 3D printers and compact laser cutters, which allow cost-effective manufacturing of small numbers of rich functional prototypes or tangibles with arbitrary forms and fine-grained structures. In recent years, such technologies have become widespread and technological advances have made them inexpensive even for technically sophisticated devices. This fact opens rich possibilities for the design and craft of tangibles and fosters tabletop TUI research for suitable application domains.

Easy and
cost-effective
fabrication of tangibles
using a 3D printer or
laser cutter.

4.2.1.3 Tangible Interaction

Tangible interaction makes use of tangibles to exploit human prehensile behaviors, human's manual dexterity and highly trained skills to manipulate physical objects with fingers and hands, and human's aptitude to act in physical space for human-computer interaction. Tangibles offer physical interaction affordances [Norman, 1999] such as bi-manual interactions, simultaneous manipulation of position and orientation, or spatial caching of objects [Fitzmaurice et al., 1995].

Benefits of tangible
interaction.

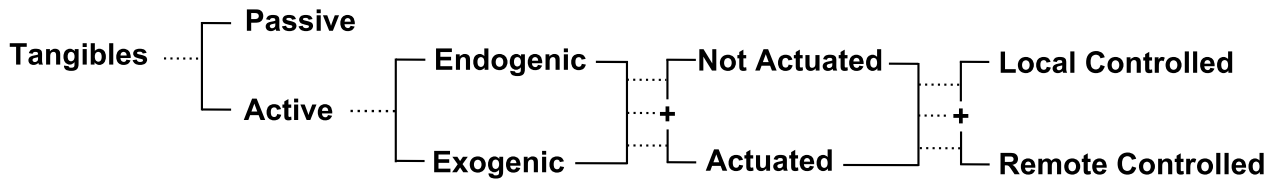


Figure 4.11: Categorization of tangible artifacts from a technical point of view.

For those physical affordances, human's natural skills are well prepared as humans are used to interact with physical objects in the real world every day.

Benefits of tabletop
tangible interaction.

Tabletop tangible interaction combines the potential of multi-touch tabletop interaction with the naturalness and intuitiveness of physical manipulation of tangibles. The rich interaction possibilities emerging from the joint usage of both technologies offer complementary as well as redundant interaction channels. For instance, tabletops afford multi-user co-located collaborative tasks through the large table-sized form factor, which perfectly complement and foster, for example, tangible learning applications for groups. However, tangibles as well as tabletop surfaces offer redundant interaction opportunities such as bi-manual, more direct, or more parallel interaction and each of them with different degrees of freedom. While tabletop surfaces limit such interactions to a 2D planar surface, tangibles enable perception and manipulation beyond the 2D planar surface.

On the one hand, consolidating multi-touch tabletop interaction and tangible interaction poses rich design choices for application developers, for example, to consider the interplay of multi-touch interactions of surfaces (with their computation and augmentation capabilities) with physical affordances of tangibles. On the other hand, however, application developers have to carefully evaluate redundant interaction channels and figure out which might be better for certain tasks with the application's aim in mind.

4.2.2 Categories of Tabletop Tangibles

How can we classify
tabletop tangible
objects?

Passive and active
tangibles.

There is a large amount of research works related to tabletop TUIs, which have developed and defined classes of tangibles over the years as technology advanced and understanding of TUIs and tangible interaction evolved. This section presents a categorization in ► Figure 4.11 for tangibles from a technical point of view. We define and describe classes and sub-classes of tangibles that were collected and consolidated based on a thorough research literature review. Tangibles can be categorized into two main classes, namely passive and active tangibles, which will be detailed in the following.

4.2.2.1 Passive Tangibles

Definition 6. *Passive tangibles are the simplest kind of physical objects, which are uni-directional manipulable by users. Such tangibles are static and inert in the sense that they cannot change their physical appearance and status by themselves nor on demand of the tabletop.*

Passive tangibles can be fabricated easily as they do not need mechanisms to reflect changes of the internal digital object state to the externalized physical representation. Due to this simplicity, most TUIs make use of passive tangibles. All tangibles except the red car in the middle shown in ► Figure 4.7 are passive tangibles.

Passive tangibles are most widespread and easy to fabricate.

The internal state of passive tangibles is conveyed solely by the tabletop, for example, by means of auditory feedback or through visual renderings and augmentations on the tabletop surface, which have its limitations. For example, if a tangible becomes spatially decoupled from the digital counterpart by lifting up through a user, then the tabletop is no longer able to link visual adaptations to the tangible due to the absence of positional and orientation information.

Limitations of passive tangibles.

Furthermore, the missing reflection of digital status changes to the physical model might leave inconsistencies between the physical and digital model [Bellucci et al., 2014, Pedersen and Hornbæk, 2011]. For instance, if an object in the digital model gets repositioned to a different location, then the TUI application has no means to express this change to the corresponding physical object in the physical world.

Inconsistencies between physical and digital state of passive tangibles.

4.2.2.2 Active Tangibles

The lack of computer controlled influence on physical objects and the potentially emerging inconsistencies between digital and physical model draw a fixed boundary for the technical possibilities and thus the design of tabletop TUIs with passive tangibles. This boundary ends where the physical state of tangibles has to be adapted in any form in order to give users feedback within the physical haptic 3D world about the current internal digital state of the object. Crossing this boundary without having computer-controllable means to act appropriately on physical objects leads to a negative impact on users' impression of a unified physical and digital interface if it not even destroys the impression completely.

Design boundary of passive tangibles.

Drawbacks of passive tangibles.

To address the limitations of passive tangibles, researchers have extended passive tangibles to active tangibles, which provide the features of passive tangibles and at the same time are able to appropriately keep the digital and the physical model consistent.

Even though the onset of research for active tangibles is dated more than 10 years ago, for example, curlybot by Frei et al. [2000] or LEGO™ robots by Ressler et al. [2001], this research field is still nascent and seems under-researched as recent works ascertained [Bellucci et al., 2014, Pedersen and Hornbæk, 2011]. The common usage of the term "active tangibles" has been established in existing research literature over the last years. Earlier works rather paraphrased such kind of tangibles with functional characteristics or descriptive terms. For instance, Frei et al. [2000] described their curlybot as a "*new class of computational toys*" and Rosenfeld et al. [2004] portrayed their active physical objects as "*movable physical objects,*" "*whereby objects can be moved both by users and by the computer*". In the context of distributed or remote collaboration, Brave et al. [1998] described the active tangibles in their PSyBench prototype as "Synchronized Distributed Physical Objects".

In the following, we give a definition for the class "active tangibles" according to the common understanding in existing research literature and go deeper into the different sub-classes by discussing the distinguishing characteristics for each sub-class.

Definition 7. *Active tangibles are capable of providing at least one kind of feedback in the physical 3D world to users controlled by a computer. In contrast to passive tangibles, such tangibles are able to "actively" reflect changes in the digital model by changing their physical model or status. Thus, active tangibles are bi-directional manipulable by users and by a computer [Rosenfeld et al., 2004, Pedersen and Hornbæk, 2011, Mi and Sugimoto, 2011a, Riedenklau et al., 2012, Bellucci et al., 2014].*

BI-DIRECTIONAL. The usage of "bi-directional" is applied slightly different in the existing research literature. Some research works relate the attribute bi-directional to the interface (i.e., the TUI [Pedersen and Hornbæk, 2011, Bellucci et al., 2014]), while others ascribe the attribute to user interaction in general [Riedenklau et al., 2012] or solely to tangibles [Rosenfeld et al., 2004].

► Definition 7 for active tangibles assumes the latter notion, which refers the attribute bi-directional to the tangibles. The former less concrete notion of

Active tangibles have been paraphrased differently in early research works.

Definition for the active tangible and its sub-classes.

Bi-directional is a certain characteristic of tangibles that actively provide feedback.

"bi-directional interfaces" would not suffice to distinguish passive tangibles from active tangibles according to ► Definition 7 because a tabletop TUI with passive tangibles (as the interface) is capable of providing feedback to users, for example, through visuals on the tabletop surface.

KIND OF FEEDBACK. ► Definition 7 does not limit the kind of feedback but allows any kind of feedback as long as the feedback is expressed through the tangible itself and controlled by a computer, which is on par with recent works [Pedersen and Hornbæk, 2011, Mi and Sugimoto, 2011b, Bellucci et al., 2014].

Kind of feedback is not limited.

Only a few works limit the feedback to be of a certain kind such as Riedenklaue et al. [2012] who defined "actuated tangibles" to incorporate physical actuation. Another example for explicitly narrowed down feedback was proposed by Inami et al. [Müller-Tomfelde, 2010, p.171] who defined active tangibles to have some form of self-propulsion, for example, using robotics or locomotion.

Few earlier works limited kinds of feedback.

Compared to these examples, ► Definition 7 is less limiting and allows various and manifold kinds of feedback. Some examples different from actuation or propulsion include tactile feedback as provided in the Pico system (Physical Intervention in Computational Optimization) of Patten and Ishii [2007] or visual/illumination feedback in the physical world such as used in "Illuminating Light" of Underkoffler and Ishii [1998].

Feedback different from actuation or propulsion.

DISTINGUISHING ATTRIBUTES. Both characteristics (bi-directionality and feedback type) create further sub-classes of active tangibles. In order to distinguish those kinds of active tangibles, ► Definition 7 implicitly bases on the following attributes that assign active tangibles to significantly different groups or sub-classes.

1. Exogenic / Endogenic Activity: Determines the origin of activities, that is, whether the activity originates from an external source or whether the activity is intrinsically originated. This attribute may also take on both values, that is, tangibles might be driven by exogenic activity and endogenic activity.
2. Actuated / Not actuated: Determines whether tangibles are motionless or make use of actuation to expose physical activity.
3. Local / Remote controlled: Distinguishes between physically distributed or only locally situated systems.

Source of activity.

Actuation and locomotion.

Physical location of control.

4.2.2.3 Exogenic and Endogenic Activity

Characteristic of the origin of an activity.

Systems that enable active tangible interfaces can be realized through two different working principles or the combination of both, which are characterized by the origin that caused the activity. The working principles differ in the origin of the cause that makes the tangibles active, that is, whether the cause originates externally, internally, or both. Depending on the working principles, the tangibles are then called exogenic⁸ active tangibles and/or endogenic⁹ active tangibles.

EXOGENIC ACTIVE TANGIBLES.

Exogenic activity is externally rooted.

Exogenic active tangibles obtain their activity from external sources due to external causes such as external laser or light illumination [Underkoffler and Ishii, 1998], a movable electromagnet under the tabletop surface [Brave et al., 1998], or an electromagnetic array mounted in or under tabletop surfaces [Weiss et al., 2010, Pangaro and Ishii, 2003, Patten and Ishii, 2007], cf. ► Figure 4.12.

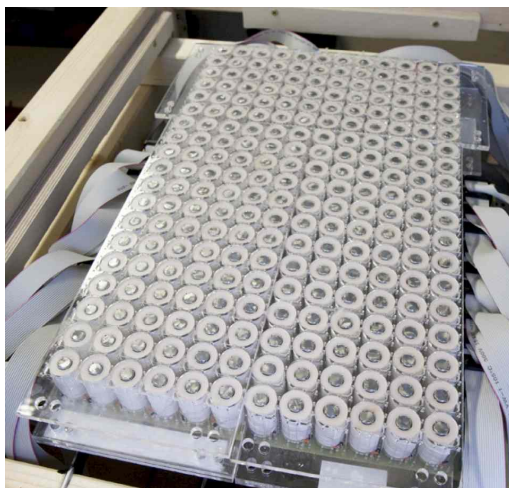


Figure 4.12: An example of exogenic active tangible systems through electromagnetic arrays: Madgets by Weiss et al. [2010].

Benefits of exogenic active tangibles.

Such tangibles are not active on their own but become active caused by external components in the environment such as the tabletop surface. Therefore, the tangibles usually do not require electrical power through batteries or power cables.

However, this approach poses many implementation related disadvantages. In case of surfaces with an electromagnetic array, for example, the tangibles may suffer from unnatural and abrupt motion [Brave et al., 1998, Mi and Sugimoto, 2011b, Pedersen and Hornbæk, 2011] or have to be endowed with appropriate magnetic responsive extensions or have to be lightweight enough to be moved by the electromagnetic array [Mi and Sugimoto, 2011b, Pedersen and Hornbæk,

⁸ Produced or originates externally.

⁹ Originating from within a system.

2011]. Apart from this, the required magnets are expensive and assembling as well as wiring the magnet array and the many micro-controller circuits require knowledge and experience in electronics, which in turn impair scalability of the approach for larger surfaces. Tabletop systems with such approaches also lose the typical (marker-) tracking capabilities of DI tabletops for tangibles on the surface due to the opaque magnet array, which is impermeable to infrared light. Such systems have to track tangibles with alternative mechanisms such as top-mounted cameras in combination with markers attached on top of the tangibles.

Drawbacks of exogenic active tangibles.

Overall, only a few systems employ the exogenic active tangibles approach due to the discussed disadvantages and high costs.

ENDOGENIC ACTIVE TANGIBLES

Unlike exogenic active tangibles, endogenic active tangibles produce feedback in the physical world by themselves, that is, without external means, for example, using LED indicators as realized by Haller et al. [2010] and Leitner et al. [2009] for the "IncreTable" or by Nowacka et al. [2013] for the "TouchBugs" (cf. ► Figure 4.13).

Endogenic activity is originated from within tangibles.

Another approach for providing visual feedback is based on embedded miniature displays showing graphics and/or texts. For this purpose, the "sifteo cubes"¹⁰ are popular off-the-shelf devices used, for example, in the "Eugenie" application by Grote et al. [2015], in the "CubeQuery" prototype by Langner et al. [2014], or the active tokens of Valdes et al. [2014].

However, the most challenging and the (physically) easiest perceptible kind of feedback employs actuation through small motors (cf. ► Figure 4.13). Therefore, many existing works target such kind of tangibles and investigated interaction approaches as well as approaches to craft and control such tangibles [Dang and André, 2013, Nowacka et al., 2013, Riedenklau et al., 2012, Pedersen and Hornbæk, 2011, Mi and Sugimoto, 2011b, Haller et al., 2010, Kojima et al., 2006, Rosenfeld et al., 2004, Ressler et al., 2001, Frei et al., 2000].

Most challenging tangibles are actuated active tangibles.

POWER / ENERGY / CONTROL CONNECTION. Due to the intrinsically generated activity of endogenic active tangibles, they require a connection to the controlling computer and have to be constantly supplied with energy in order to drive, for example, control logic, displays, LEDs, loudspeakers, or motors. This can be realized either cable-connected (through slim wires) or wireless (battery-powered). Using a wired connection is the easiest approach, which usually supplies the tangibles with power as well as control connection to the tabletop computer. Wired connections are most suitable for tangibles that do not have to move over the surface as wires disturb and constrain possible movements. However, early work such as the "Augmented Coliseum" of Kojima et al. [2006] made

Wired tangibles are easy to craft but may limit movements.

¹⁰<https://www.sifteo.com/cubes>

Figure 4.13: An example of tangibles that provide active tactile feedback to users: Touchbugs by Nowacka et al. [2013].



use of cable-connections due to technological possibilities and limitations of that time.

Most recent works supply tangibles with energy by means of standard rechargeable or off-the-shelf embedded batteries, for example, as demonstrated by Mi and Sugimoto [2011a]. Such tangibles often employ integrated micro-controller boards (e.g., based on Arduino¹¹ or Atmel ATmega¹²) and are tethered to tabletop computers through a wireless connection (e.g., XBee RF modules¹³).

EXOGENIC VS. ENDOGENIC. In systems with endogenic active tangibles, the technical complexity is located in the tangibles instead of the tabletop hardware. This poses a big advantage over systems for exogenic active tangibles because of the much better portability of the tangibles to other existing tabletop platforms and thus the opportunity to reuse tangibles for multiple different systems and applications. Furthermore, systems for endogenic active tangible provide inexpensive and much better scalability to larger surfaces, as the generation of activity does not depend on particular tabletop hardware implementations. In contrast, the cost of scaling of exogenic active tangible systems to a larger surface is disproportionately higher as the magnet array as well as the complete control logic has to be extended and reprogrammed.

It is imaginable that both mechanisms, that is, endogenic and exogenic, are implemented by the same tabletop system. However, to our knowledge, no one has presented a system that realize this approach.

¹¹<http://arduino.cc>

¹²<http://www.atmel.com/products/microcontrollers/avr>

¹³<http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module>

Endogenic active tangibles are better reusable.

Endogenic active tangible systems are better scalable.

4.2.2.4 Active Actuated Tangibles

A significantly distinguishing attribute of active tangibles pertains to whether they are actuated or not because of the physical properties that an adaptation changes, such as position, orientation, motion or even form and shape. Those physical properties benefit from humans' visual perception abilities to quickly recognize moving objects in the visual scene.

Benefits of actuated active tangibles.

HUMAN VISUAL PERCEPTION. Research in the discipline of neurobiology studied and identified a certain visual area called V5 or MT within the visual cortex of human brains that is dedicated to the processing of motion and depth in the visual field [Born and Bradley, 2005]. Explanations for those human skills emanate from the assumption that humans have developed highly specialized skills to detect moving objects over millions of years of evolution since quickly detecting enemies and predators was constantly necessary for survival [Ware, 2008, p.36-38]. Therefore, we can assume that users perceive interfaces using actuated active tangibles different from motionless active tangibles.

Humans have specialized skills to perceive moving objects.

IMPLEMENTATION APPROACHES. There are different approaches to realize actuated active tangibles ranging from retrofitting off-the-shelf available actuated objects [Pedersen and Hornbæk, 2011] or toys [Dang and André, 2013] to constructing and building custom active tangibles. Most researchers crafted self-made actuated active tangibles, for example, by means of LEGO's MindstormsTM play pieces¹⁴ [Ressler et al., 2001], by assembling and reprogramming available components [Riedenklaus et al., 2012], or completely from scratch [Mi and Sugimoto, 2011b, Kojima et al., 2006, Rosenfeld et al., 2004, Frei et al., 2000]. In order to enable actuation, tangibles are endowed with one or more miniature motors to actuate mechanical parts of the tangible [Mi and Sugimoto, 2011b] or to drive wheels, which physically moves and rotate the tangibles [Kojima et al., 2006, Rosenfeld et al., 2004, Frei et al., 2000].

Different approaches to craft actuated active tangibles.

Actuation enabled through miniature motors.

CONTROLLED ACTIVE TACTILE FEEDBACK. Actuation is also employed to provide controlled tactile feedback to users [Mi and Sugimoto, 2011b, Pedersen and Hornbæk, 2011, Patten and Ishii, 2007]. In this case, tangibles usually have to be enriched with feedback sensors that detect when users perform force on the tangibles. Tactile feedback may help users to perceive different steps on a height-adjustable scale, such as those in the HATs system [Mi and Sugimoto, 2011b], or to indicate the start and end of a scale by providing appropriate tactile resistance through the motors [Pedersen and Hornbæk, 2011].

Tactile sensors to enable controlled actuation as feedback.

¹⁴LEGO Mindstorms. <http://www.legomindstorms.com>

4.2.2.5 Local and/or Remote Controlled

In terms of user interaction, interfaces based on active tangibles receive user input through physical manipulation of the tangibles and provide system output through the physical activity of tangibles controlled by tethered tabletop computers. Considering this bi-directional character of active tangible interfaces and in particular the ability to adapt the physical state of tangibles, it is consequent to extend such systems for distributed collaborative computing through inter-connecting multiple active tangible systems. Such a distributed system allows synchronizing the digital model across all connected tabletop computers, which in turn (magically) adapt the physical models of all coupled active tangibles (on all connected tabletop computers). Hence, users are able to collaborate with remote users by means of active tangibles, not only on the 2D surface but also using the physical space above the tabletop surface.

Connecting multiple actuated active tangible systems together enables remote active tangible interfaces.

Similar to active tangible research in general, distributed active tangible interfaces seem under-researched, thus only a few works focusing on this topic are available. Though, interest has arisen quite early in 1998 when Brave et al. [1998] presented their concept of "Synchronized Distributed Physical Objects". One of their prototypes called PSyBench (Physically Synchronized Bench) was intended to allow distant users to cooperatively manipulate and consider the resulting illuminated light scenes caused by the active tangibles. Based on their concept, they discussed issues such as level of synchronization, social phenomena (e.g., tangible presence), or interaction related issues (e.g., co-located vs. distributed users).

Early research investigated technical as well as social issues of remote active tangible interfaces.

Richter et al. [2007] and Müller-Tomfelde [2010, p.177] used the terms local and remote active tangible interactions to describe systems that are either distributed across multiple tabletop displays or operating merely on one tabletop display. More recently, Riedenklau et al. [2012] employed technologically advanced components to build so-called actuated TAOs (Tangible Active Objects) in order to realize a distributed collaborative planning application.

4.3 A Study on Multimodal Active Tangible Interaction - TabletopCars

Contribution Statement: In this section, we contribute to an under-researched domain, which combines multimodal tabletop interaction with active actuated tangible interaction. We present a system, concept, design, and evaluation that address this research domain. An earlier version of this section's content has been published as a peer-reviewed full paper [Dang and André, 2013] at the 7th International Conference on Tangible, Embedded and Embodied Interaction, TEI'13.

<https://doi.org/10.1145/2460625.2460630>

So far, we have reviewed and discussed multimodal interaction and tangible interaction in the context of tabletop computing separately. Even though tabletop tangible interaction can be seen as part of multimodal tabletop interaction by definition, the possibilities and the significance of tabletop tangible interaction make this topic an outstanding and separate research area in comparison with other fields of multimodal tabletop interaction. This fact becomes apparent when comparing the quantities of work for both topics in the existing research literature related to tabletop computing.

A substantial number of publications for tabletop TUIs is available and even a dedicated conference for tangible interaction (TEI - Tangible and Embedded Interaction), which includes tabletop TUIs, has been established since 2007. In contrast, far fewer works for other modalities in combination with tabletops, such as speech, free-air hand gestures, full-body gestures, or eye gaze, were published. However, there is a large body of multimodal interaction research apart from tabletop computing, which shows great potential that can be exploited for tabletop computing.

MULTIMODAL ACTIVE ACTUATED TANGIBLE INTERACTION. In summary, both discussed topics offer compelling opportunities to realize tabletop interaction beyond touch. For example, active actuated tangibles provide feedback in the physical 3D world while multimodal tabletop interaction enables input above or around the digital tabletop. Both realize tabletop interaction beyond touch by their own. Moreover, technological advances in recent years in terms of miniaturization of active tangibles and novel sensors (e.g., depth sensors), (1) foster the possibilities for novel prototype systems, (2) broaden the design possibilities for

Multimodal tabletop interaction vs. tabletop tangible interaction.

Combining multimodal tabletop interaction and tabletop tangible interaction.

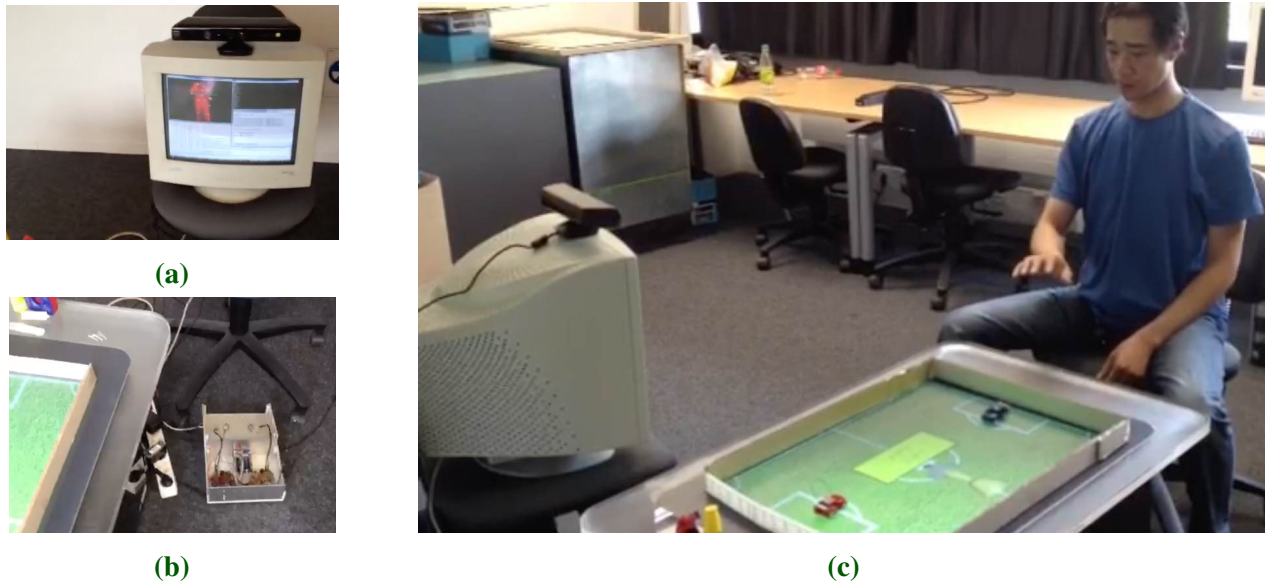


Figure 4.14: A test scenario in the lab during development: (a) Depth sensor on the monitor that sense users in front; (b) The remote controller unit connected to the tabletop system; (c) A user performing gestures to control the blue racecar on the tabletop surface.

interaction techniques, and (3) might lead to enhanced user experience for tabletop interaction beyond touch. Therefore, we combined both worlds and present a case study that brings together active tangibles and embodied interaction as a natural form of multimodal interaction.

TABLETOPCARS. In the remainder of this section, we discuss a case study for multimodal active tangible interaction, which includes the design and implementation of the prototype called TabletopCars and a first study that explored the realized interaction techniques. We enabled users to control small actuated racecars on a tabletop surface through gestures performed with arms and hands in the air, cf. ► Figure 4.14c. Building systems to investigate such kind of interaction usually requires laborious handicraft work to fabricate the active tangibles [Kojima et al., 2006] or to establish reliable position sensing of the tangibles such as proposed in the system "Display-Based Measurement and Control System" by Sugimoto et al. [2007]. The case study in this section presents an easy and low-cost approach to realize multimodal active tangible interaction based on microsizeers. User input performed by the user's body was sensed by means of an off-the-shelf depth camera as shown in ► Figure 4.14a and translated into instantaneous commands to control the small racecars. In order to wirelessly control the small actuated cars, we built a remote controller unit as depicted in ► Figure 4.14b, which translated user's commands into appropriate radio-frequency signals that the cars understood. In terms of the application domain, TabletopCars was designed as a tabletop game due to its appropriateness for small racecars.

What is TabletopCars?

TabletopCars demonstrates an easy and low-cost approach to build TUIs with active actuated tangibles.

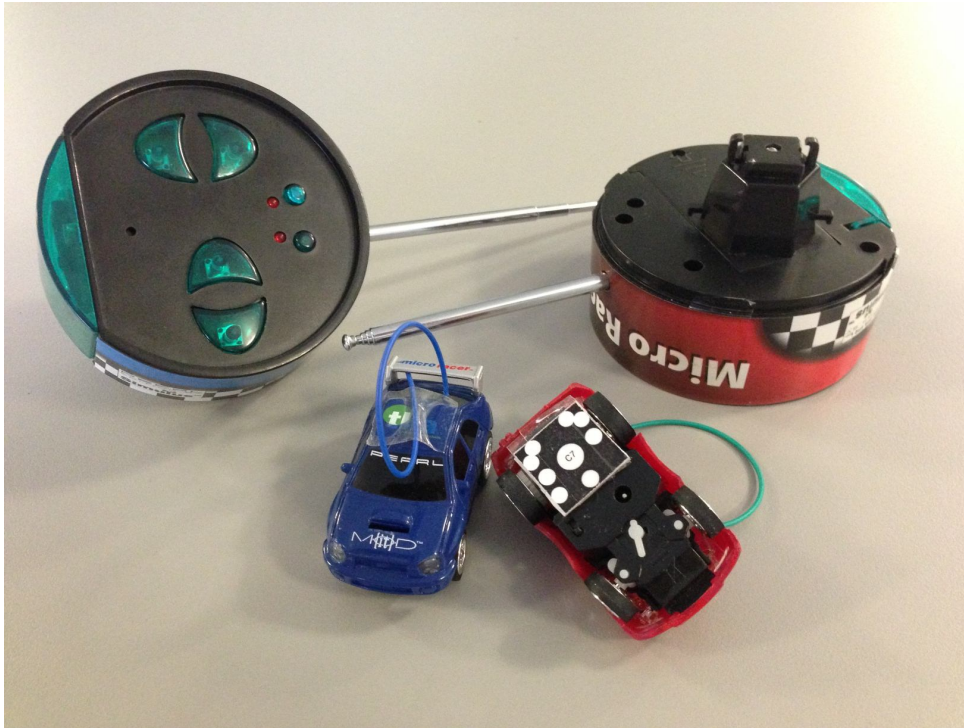


Figure 4.15: The microsizers together with according radio remote controls. Each microsizer has a byte tag mounted on the underside.

4.3.1 Design Rationale: Why R/C cars and why tabletop games?

Small-scale radio controlled cars, also called R/C cars, have always been fascinating and attractive toys for all age groups. R/C cars have in common that they are controlled over a radio-frequency link with a wirelessly connected controller device.

Short overview of R/C cars and its application.

There is a variety of different kinds of R/C cars, which vary along the cost, size, power source and engine, achievable speed, or level of detail in terms of resemblance to a real car model. In general, such cars can be electrically powered or nitro-/gas-powered. Nitro powered R/C cars are meant to be operated outdoors, for example, at large race courses where multiple players compete for the best lap time or to be the first player who crosses the finish line. Actually, the most frequent usage for nitro-powered cars are competitions that are organized as professional events by R/C car clubs. Electrically powered models, however, can also be operated within buildings or small rooms since they do not produce exhaust fumes.

The sizes of electrically powered models range from scales of 1:5 down to 1:87 or even smaller, whereby the smallest models are called microsizers with lengths around 5cm (cf. ► Figure 4.15). Such models often have limitations, for example, in terms of power capacity or radio control capability. While the bigger sized models have to be operated on large areas, the application of microsizers is

The smallest kind of R/C cars is called microsizer.

Limitations of
microsizers.

Area coverage
limitation of
microsizers is
beneficial for
tabletops.

Potential for
microsizer
applications through
combining microsizers
with digital tabletops.

Local endogenic
active actuated
tangibles.

limited to small areas due to their weak radio-frequency link. The radio control's coverage of such microsizers is usually up to 10m, which makes them well suited for games that require only a little space. An example of such a requirement is driving the microsizers on a table where players are located face to face. This quality is inherently different from the typical car-racing scenario where players usually stand side by side and concentrate on distant R/C cars. The proposed scenario offers chances and qualities known from board games.

Instead of playing on a static table surface, employing an interactive tabletop such as the Microsoft Surface [Microsoft, 2011] for games with microsizers offers much potential for dynamic enhancements. In comparison to a static table, interactive tabletops enable to display arbitrary digital content on the tabletop surface as well as tracking any kind of objects or contacts on the surface. Hence, combining microsizers with interactive tabletops provides rich possibilities for creating game designs, novel game concepts, and increased game experiences, for example, through integrating virtual and real world objects into the gameplay as proposed by Haller et al. [2010].

Furthermore, this concept benefits from qualities that are known from computer games, such as computer-mediated game management or augmentation of the microsizer depending on their playing context.

KIND OF INTERACTION IN THE DESIGN SPACE. Interaction techniques for small remote controlled cars on a tabletop surface fall into the general category of interaction with tangible user interfaces as discussed in ► Section 4.2, whereby the usage of tangible objects that are self-propulsive in the form of locomotion assigns the tangibles into the concrete group of actuated active tangibles (cf. ► Subsection 4.2.2.2). Furthermore, the tangibles are endogenic local active tangibles since the microsizers are controlled by only one tabletop system and activity is generated by the tangibles themselves. ► Figure 4.16 shows (accented

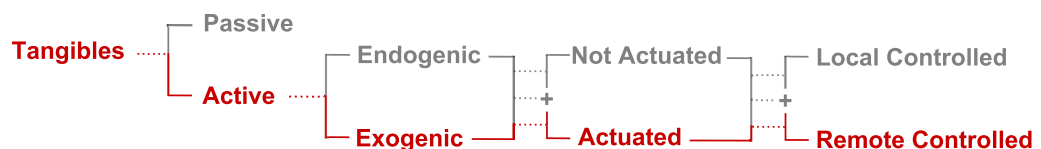


Figure 4.16: Tangibles of TabletopCars in the categorization of tangible artifacts.

with red color) how the tangibles in TabletopCars is mapped in the classification of tangible of ► Subsection 4.2.2.2. The kind of interaction in TabletopCars is represented as the combination of embodied interaction with tangible objects (i.e., Row 4 and row 8 of ► Table 4.1). We have replicated the important rows of the design space in ► Table 4.2.

	Interaction	Modality	Medium	Code – Applications
3	In-the-Air / Above surface gestures, pointing	Visual, Tactile, Kinesthetic, Proprioception	Camera (I), Motion capture system (I)	Pose / Gestures with head, body, hand, Arm, finger; Pointing / Location – User mapping, Pick-and-Drop
4	Bodily, Embodied	See row 3	See row 3	See row 3 + row 5
8	Tangible Object	See row 3	Physical artifacts (I/O)	Location, Pressure, Force

Table 4.2: TabletopCars interactions in the classification of multimodal tabletop interaction.

BUILDING BLOCKS. The design proposed in this section combines microsizers with an interactive tabletop, thus extends the range of applications for microsizers and at the same time offers a testbed for investigations for multimodal active tangible interaction. The implemented TabletopCars prototype consists of four different games for the microsizers. Furthermore, we connected the original radio control units of the active tangible cars to our developed software layer by utilizing a microcontroller. In the following, we describe the whole system in detail and report on issues that arose in engineering such an interactive system. Finally, we present results of a preliminary user study that indicates the general acceptance of multimodal active tangible interaction based on microsizers.

4.3.2 Related Work

Games that are based on tangible user interfaces and interactive tabletops have often been the focus of investigations, for example, in STARS [Magerkurth et al., 2004], Weathergods [Bakker et al., 2007], PINS [Kirton et al., 2008], IncreTable [Leitner et al., 2010], Optical Chess [Wu et al., 2010], Comino [Leitner et al., 2009], or Futura [Speelpenning et al., 2011].

However, only a few works address active tangible interaction. On a conceptual level, Jain et al. [2006] presented Sketch-a-Move, which allowed children to explore the relationship between courses, that were drawn by the finger on small physical cars, and the corresponding physical movements of the cars. In comparison to the concept of TabletopCars, Sketch-a-Move involved separated interaction and execution phases, whereas the interaction in TabletopCars is direct and immediate.

Robert et al. [2011] built a mixed reality robot gaming platform in which the user tele-operated a small robot called Miso. This robot was the active tangible, which moved within a hybrid space that combines physical and virtual spaces by means of displays and projectors. The projection was used to augment the

Most research works on tangible tabletop games do not consider active tangibles.

The concept Sketch-a-Move.

Active tangible robot interaction with virtual entities.

physical space with virtual objects in order to create a mixed reality. Their mixed reality system focused on the interaction of a robot with its virtual peers, whereas we address novel interaction possibilities with active tangible microsizers.

Active tangible robot tracking.

Kojima et al. [2006] reported on an augmented game environment with small vehicles that is called augmented coliseum where a robot attacked another robot with a laser cannon that was projected into the game environment. Their augmentation of the game environment also included a central part of the tracking system called display-based measurement system, which was the focus of their work.

Active tangible small cars without tabletops.

Tanev et al. [2006] and Tanev and Shimohara [2007] presented a system that connected the radio remote control of a small-scale car to a computer. They tracked the position of the microsizer by means of a live video feed of the gaming environment and developed a driving agent that remotely operated the car through a lap with obstacles. Tanev's work focused on algorithms that enabled the driver agent to optimally achieve the best lap time in a manner that competes with human drivers.

System design of active tangible Lego™ robots. No user interaction investigated.

Robot Arena by Calife et al. [2009] is a system that built on small Lego robots with an interactive table. The position of the robots on the table surface was tracked via a top-mounted webcam combined with markers on top of the robots. The presented system represents an infrastructure for the development of novel games and interactive applications. Therefore, the authors concentrated on describing the system design and architecture instead of interaction possibilities and modes.

Virtual racecars on tabletops controlled with gamepads.

The work of Haller et al. [2010] reported on design recommendations concluded from experiences with several tabletop games. One of the games was called NeonRacer, which is related to TabletopCars in the sense that small cars had to be navigated through a course with tangible obstacles. They showed that everyday objects such as beverage cans or cups can be used to enhance the gaming experience on interactive tabletops. In contrast to TabletopCars, the cars in NeonRacer were virtual objects and the interaction was carried out with traditional gamepads. The authors found in a pilot study that players often had orientation difficulties caused by the car orientation when using the gamepad. Many participants would prefer a more intuitive interface for the interaction. Their results motivate TabletopCars, which offers the possibility to connect natural interaction devices such as the Microsoft Kinect depth sensor to realize interaction.

Embodied interaction increases player's engagement.

Furthermore, supporting embodied interaction also increases player's engagement as found by Bianchi-Berthouze et al. [2007] or Lindley et al. [2008], which gives promise for novel gaming experiences through tabletop games such as TabletopCars.

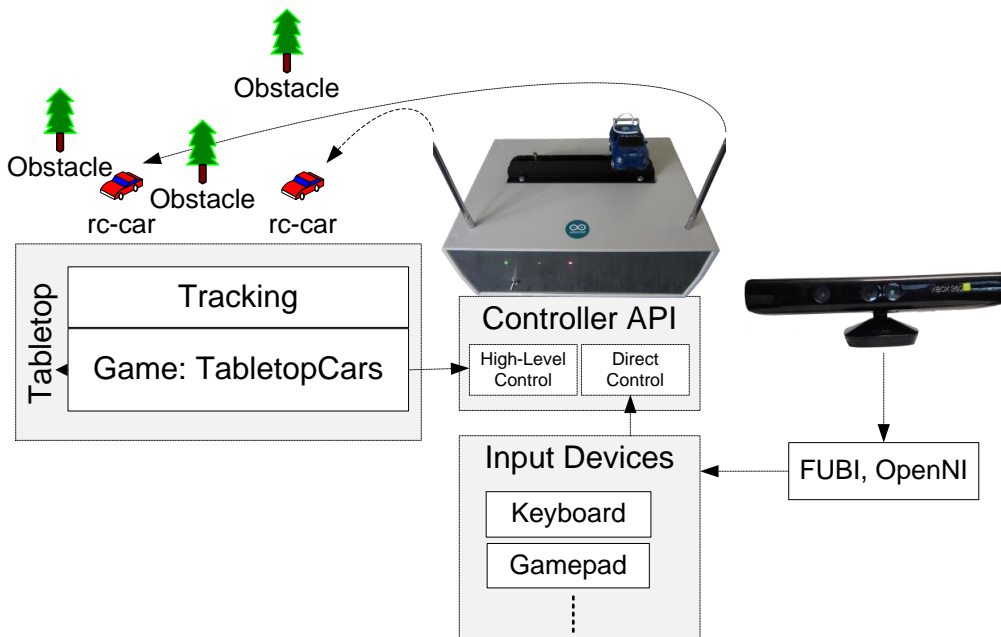


Figure 4.17: Outline of the TabletopCars system with controller, sensors, obstacles, and according software components.

4.3.3 System Outline

To realize our active tangible game system, we combined the Microsoft Surface tabletop with micro-sized cars from Stimulus as sketched in ► Figure 4.17. In order to enable embedded and embodied interaction experiences, we built a controller unit that establishes wireless control of the micro-sized cars through arbitrary input devices and systems, such as the Microsoft Kinect sensor, a standard PC-keyboard, or traditional gamepads. The following sections describe each of the components in detail.

4.3.3.1 Microsoft Surface

TabletopCars was implemented for the Microsoft Surface tabletop, which is a commercially available horizontal multi-touch tabletop that provides a development kit for the creation of applications. The form factor of the tabletop conforms to a living room table where people sit around on chairs or couches. The tabletop's vision system offers multi-touch recognition and object tracking based on infrared light technology and rear-projection. The tabletop surface measures 24" x 18" with a resolution of 1024 x 768 pixels.

4.3.3.2 Micro R/C Cars

The micro-sized cars (Stimulus NC-1195/NC-1196) employed in TabletopCars are miniature models of racecars with a size of 61mm x 32mm x 28mm (L x W x

Micro sized R/C cars can turn left/right and drive forward/backward. Battery lasts for about 5 minutes.

H). Their engine, magnets for steering control, and lights are electrically powered by a rechargeable battery. The battery can be fully charged within 10 minutes and lasts for about 5 minutes of continuous operation. The engine and steering mechanics are radio-controlled over a wireless connection at the frequencies 27 MHz and 40 MHz by means of a proprietary frequency signal scheme.

The original radio control unit depicted in ► Figure 4.15 provides four buttons for controlling the car. The buttons at the top and bottom produce forward and reverse motion on the back wheels at a constant speed, whereas the buttons at the left and right produce left and right turns on the front wheels. Thereby, these commands serve to drive the car forward or backward at a constant speed while having the choice to drive straight or to turn left or right at a fixed turn radius.

POSITION AND ORIENTATION SENSING. In order to realize a game management that maintains a state of the physical world above the tabletop surface, all tangible objects needed to be tracked and identified by the system. For this aim, TabletopCars made use of the tabletop vision system's capabilities that include detection and tracking of so-called byte tags¹⁵, which are also known as surface domino tags. Such tags encode a byte value (0 to 255) and also the direction in which the tag was placed by means of a geometrical scheme. In the TabletopCars system, a byte tag was mounted on the underside of each car in between the rear wheels as depicted in ► Figure 4.15. The gap between the tabletop surface and the byte tags were adjusted to about 1mm in order to ensure a stable detection and to prevent friction between the byte tags with the tabletop surface.

Tracking of cars employed Microsoft Surface byte tags.

Speed of cars had to be reduced by means of a resistor.

Furthermore, the motor speed of the cars in their original condition was too high for the tabletop's vision system, resulting in losing track of the cars at default driving speed. Therefore, we electrically modified the cars and reduced the motor speed by installing a resistor in between the main power line of the car engine and the battery. This modification reduced the maximum speed of the cars as well as the start-up speed while maintaining full functionality. The resistor had a value of 15Ω , which was the best trade-off between slowing down the cars while ensuring a reliable operation of the engines. Despite this modification, the cars might still drive too fast for a short time in the rare situation when a car drives over the full distance of the surface without stopping. TabletopCars compensated this rare situation successfully by allowing short periods ($\leq 1\text{sec}$) of tracking loss.

¹⁵<http://www.microsoft.com/download/en/details.aspx?id=11029>

4.3.3.3 Physical Obstacles

In addition to the micro-sized cars, we employed physical objects as part of the game environment in TabletopCars. The objects acted as obstacles, which the

3D printed obstacles as passive tangibles.

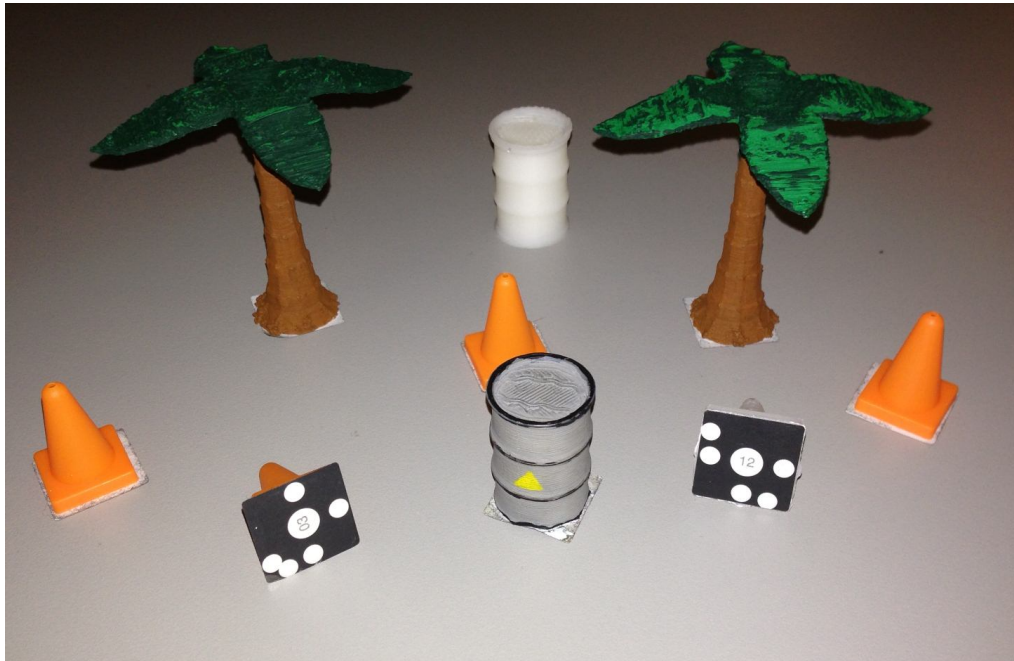


Figure 4.18: The tangible objects used as obstacles in TabletopCars: miniature models of trees, traffic cones, or oil-drums. Each obstacle has a marker glued to the underside.

cars had to drive around, for example, miniature models of trees, traffic cones, or oil-drums as depicted in ► Figure 4.18. These objects were modeled in Blender¹⁶ (an open source 3D content creation suite) and printed by means of the BFB-3000¹⁷, which is an affordable desktop 3D printer. The objects were assembled in layers, which consisted of either PLA (Polylactic Acid) or ABS (Acrylonitrile Butadiene Styrene) thermoplastic material with a layer thickness of 0.125mm. This layer thickness enabled printing of detailed and fine grained structures as can be seen in ► Figure 4.18 (e.g., the oil-drums).

The printed objects were painted afterwards to match the coloration of real objects. In order to achieve optimal colorings, they were printed with white colored material as white material affords the most neutral background. Finally, the objects were tagged with a byte tag at the bottom side in order to be tracked by the tabletop system.

Another approach to realize tracking is to consider the pattern of the byte tags in the design stage, that is, to include the pattern on the bottom side of the model and print the pattern as part of the object, which also works well. However, we decided to use additional (paper) byte tags due to a higher flexibility in case of changing the tag values for particular objects.

Designing markers as part of the tangibles to be 3D printed.

¹⁶<http://www.blender.org>

¹⁷<http://www.bitsfrombytes.com>

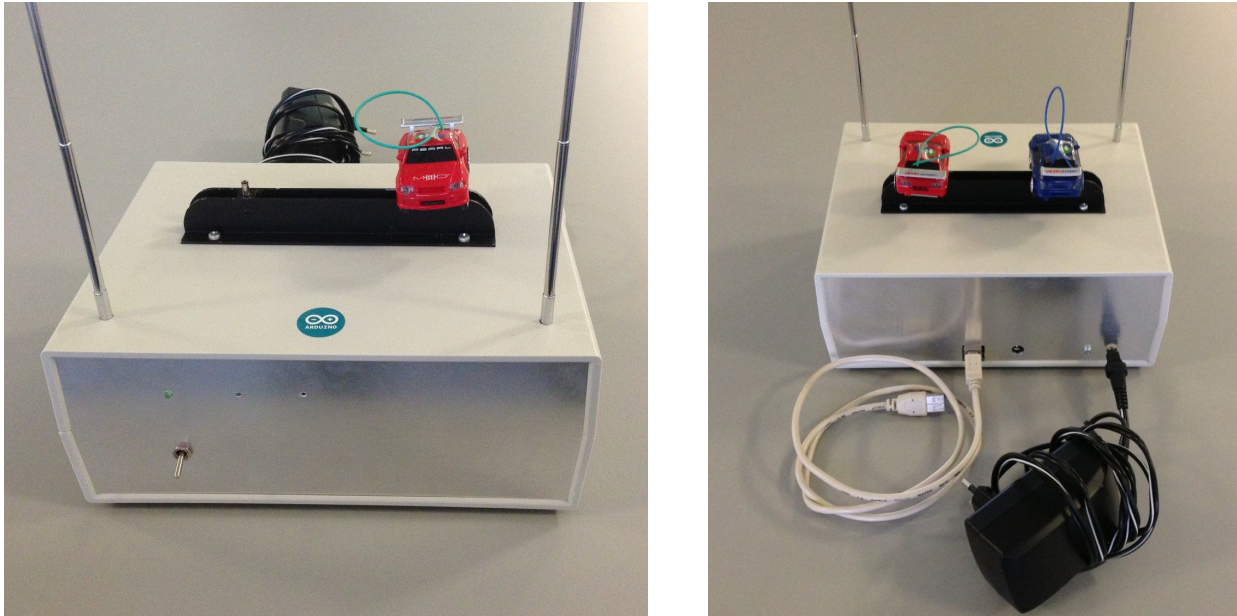


Figure 4.19: The front- and backview of the remote controller unit of TabletopCars.

Overall, the proposed process to create physical objects is ideal for realizing tangible objects for tangible interaction, since it enables to model individual objects easily while taking haptics into account. Furthermore, it allows for cost-efficient producing of individual designs that are customized to particular applications.

4.3.3.4 Remote Controller Unit

In order to build the basis for realizing embedded and embodied interaction, the micro-sized cars needed to be controllable by the tabletop computer. Therefore, we developed a remote controller unit that connected the original remote controls of the cars to a serial port by means of a microcontroller. The remote controller unit depicted in ► Figure 4.19 also housed the antennas, charging indicators, and served as a charging station for the cars. Furthermore, an external power supply made the system independent of additional batteries.

The remote controller unit was built on the open-source rapid-prototyping board Arduino Uno Revision 3¹⁸, which is based on the microcontroller Atmel 328 as the core component. The Arduino platform is widely used for physical computing development due to its simple development environment that employs a variant of the C programming language accompanied by an easy-to-use, rich, and versatile function library. Hence, it enables development of prototypes in quite a short time and supports quick experimentation cycles with less effort.

¹⁸<http://arduino.cc/en/Main/ArduinoBoardUno>

Arduino based
controller unit with
charging stations.

Arduino platform
widely used for
physical computing.

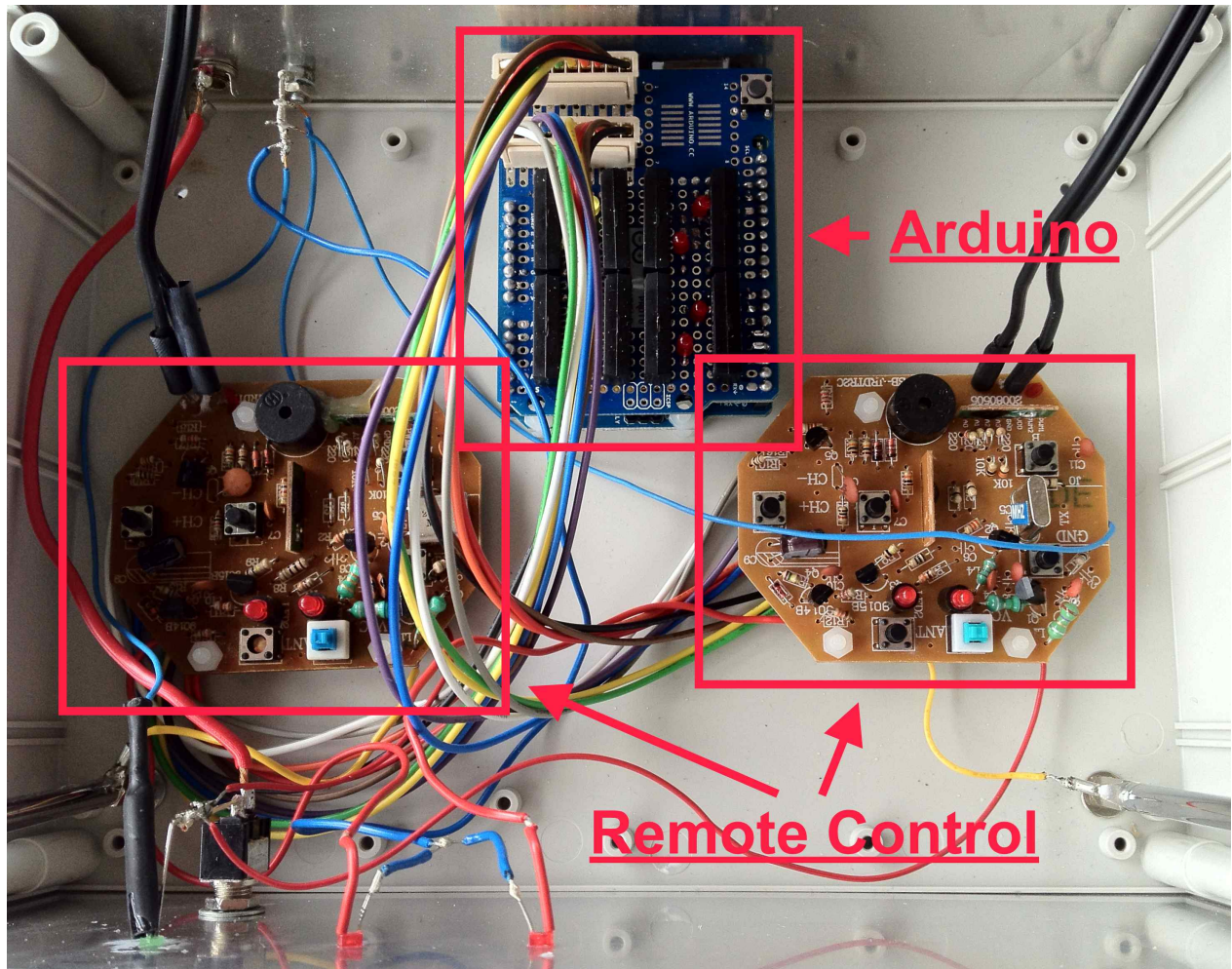


Figure 4.20: The inner view of the remote controller unit with the microcontroller in the middle and the original remote controls at the left and right.

The microcontroller in TabletopCars provided 14 digital input/output ports, of which eight were used to drive reed-relays. In turn, the reed-relays triggered button presses on the original remote controls (cf. ► Figure 4.20) in order to submit the command wirelessly to R/C cars.

CONTROLLER API LAYERS. In order to keep the remote controller unit's operation as stable and reliable as possible, the functionality within the microcontroller's firmware was limited to the basic commands, that is, drive forward, drive backward, turn left, and turn right. In addition to that, we developed an extended drive command, which produced drive pulses similar to a PWM (Pulse Width Modulation) to realize different driving speeds. The remote controller unit provided a USB connection that offered a virtual COM port for communication with TabletopCar's high-level API.

Controller implemented basic car functions (drive and turn) and extended drive pulses.

High-level API
enables direct control
of car functions.

High-level API enables
redirect and modify
car commands before
sending to a car.

TabletopCar's high-level API was developed as a C#-library that wrapped the connection and low-level communication with the microcontroller into a static class. This class enabled the submission of direct commands to the cars for driving or steering, which made it simple to map input devices such as a keyboard, a WiiMote controller, or a traditional gamepad directly to the cars' functions. For instance, gamepad buttons or keys (chosen by users) can be directly mapped to steering commands. Each steering command might also be redirected through a game logic to influence the behavior of a car, for example, to match certain game conditions such as imitating the behavior of a car when driving over a sleek or cluttered ground. Furthermore, the library offered higher-level functions such as driving to a given position, which involved submitting a sequence of driving and steering commands depending on the current tracked position for the according car.

4.3.4 TabletopCars Games

Four competitive
games.

TabletopCars was developed as a C#-application employing the Microsoft Surface SDK and provides four different game modes, which have in common that they have to be played in a competitive manner. Each of the game modes has a time limitation that requires players to gain as many points as possible within the time frame. In order to start a game, the players' corresponding cars (i.e., their byte tag value) have to be identified. Hence, each car has to be placed at a defined start location as depicted in ► Figure 4.21. As soon as TabletopCars has detected the byte tag values and mapped them to the corresponding cars, a timer counts down to zero. The players then have to achieve the goal of the particular game.

Games pause if
tracking of a car is
lost.

The bridge between the virtual and physical world is based on the tracking capabilities of the tabletop system, which can be broken in certain situations, such as when a car leaves the tabletop surface, or when obstacles get knocked over by a car. In such cases, TabletopCars pauses the game and indicates the cars' last positions in the virtual world by displaying blinking circles at the particular locations on the surface. In order to continue the game, the cars have to be placed at the correct locations after which a countdown starts again.

4.3.4.1 Car Soccer

Control car to kick a
soccer ball into the
opponent's goal.

Similar to a real soccer game, the Car Soccer game offers a green meadow with big goals on both sides of the play area as depicted in ► Figure 4.21. Furthermore, there is a virtual soccer ball that can be moved over the play area. The aim of this game is to achieve points by kicking the ball into the opponent's goal.



Figure 4.21: Car Soccer game: The initial placement of players' cars. Tracked positions are visualized with colored circles.



Figure 4.22: Car Crashing: The blue car hitting the red car at the side.

TabletopCars takes the dimensions of the cars into account for a simple collision detection to recognize when a physical car gets in contact with the virtual ball. Depending on the speed of the car and the point of contact, the virtual ball gets either moved or kicked in the appropriate direction, thus gives the virtual ball a realistic behavior that imitates a real soccer ball in terms of physical behavior. Furthermore, a sound for kicking the ball and scoring a goal gives additional feedback to the players.

Simple collision detection that includes the size and speed of cars.

4.3.4.2 Car Crashing

In contrast to the Car Soccer game, the Car Crashing game focuses on physical contact of the cars. The aim of this game is to drive with the front of the car in the opponent's car. Points are distributed depending on the parts of the cars where the hit took place. If both cars hit each other in the front, then both players

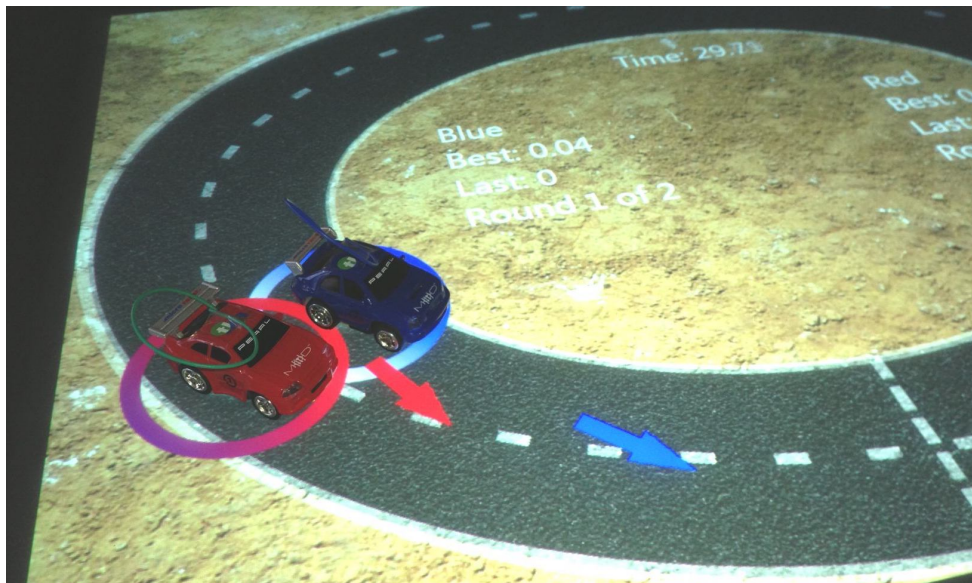
score 3 points because none of the players is at an advantage. If one player hits the other at the side (cf. ► Figure 4.22) or at the back of the opponent's car, then the player causing the hit scores 5 points because the player is at an advantage and the active part within the crash.

4.3.4.3 Fastest Lap

One of the most common games with remote controlled cars is driving a car on laps as fast as possible in order to achieve the shortest time for a lap. Fastest Lap implements this game on the tabletop surface where players have to skillfully maneuver their cars as quick as possible through a lap over the track.

Figure 4.23:

Fastest Lap:
Two-player mode
with both cars on
their according
track.



The game area consists of a circular track (on which both cars have to drive) together with the start and finish line. Since there is no physical barrier that delimited the round track, the cars can also drive off. Therefore, colored arrows correlating to the colors of the cars assist the players and indicate transit points on the track that needs to be crossed by the cars in order to successfully complete a lap (cf. ► Figure 4.23). For each lap, the lap times are measured and shown together with the best lap time per player. Fastest Lap also offers to change the number of laps for each game session within an options menu. The player with the best overall lap time wins the game. Fastest Lap offers a single player mode and a two-player mode as depicted in ► Figure 4.24 and ► Figure 4.23. In contrast to the two-player mode, the single player mode records the actual path that the car drives for each lap and stores the path for the best lap time. Thereby, a virtual ghost car that displays the path of the best lap in the correct chronology accompanies the game as soon as a recorded lap is available.

Cars might drive off
the tabletop.

Single- and two-player
modes.

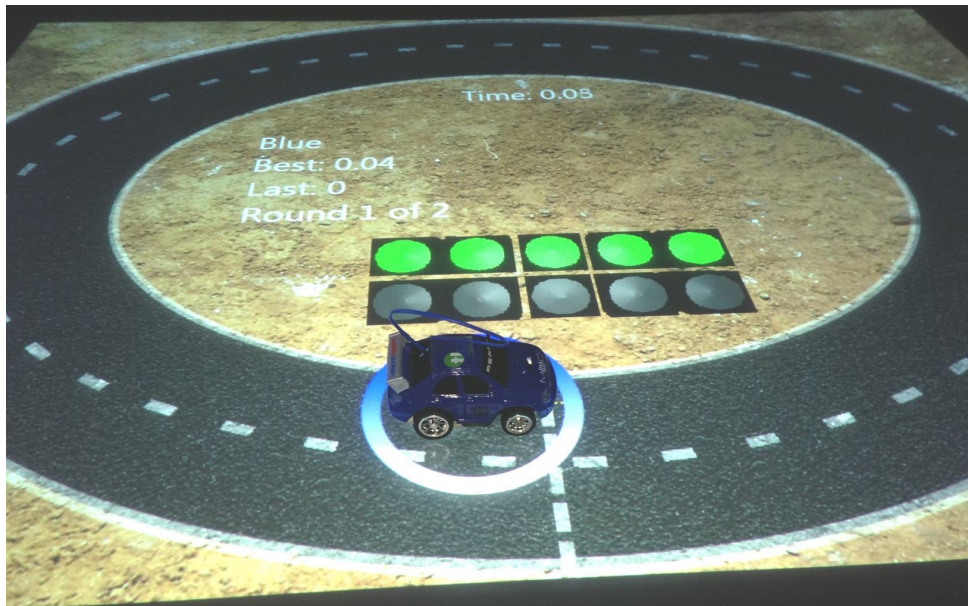


Figure 4.24:
Fastest Lap:
Single player
mode with only
one car.

4.3.4.4 Parcours

The Parcours game makes use of additional tangible objects as obstacles (cf. ► Figure 4.18). They are placed on the tabletop surface to compose a course, which the cars have to drive through as exemplarily depicted in ► Figure 4.25. Parcours offers two modes, of which the first one provides a level editor to create courses with particular obstacle placements, and the second one provides the actual game to play a formerly determined course.

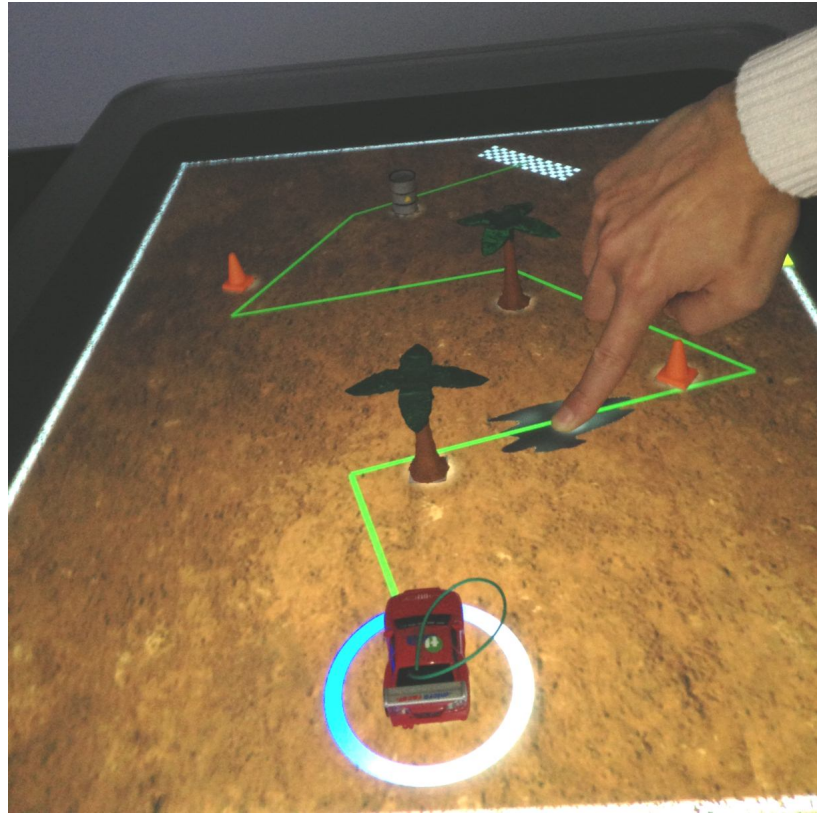
GAME RULES. In order to successfully pass a course, players have to maneuver their cars through several waypoints as quick as possible and with as few mistakes as possible. The waypoints are connected to a path that is visually indicated by green lines, as shown in ► Figure 4.25. Each course has a start and finish line that needs to be passed by the cars whereby they trigger a stopwatch, measuring the time taken to pass the course. In addition to the physical objects on the surface, there are virtual oil slicks on the course that the cars have to drive around. In order to determine the winner of a match, the game manages a score for each player, which is calculated as follows. Each passed obstacle scores 10 points and each collision with an obstacle reduces the score by 10 points. In case a car hits an oil slick, the car's score falls off by 5 points. Finally, the score is weighted by the time taken to pass the course in order to determine the overall score.

LEVEL EDITOR. The level editor enables a player to compose a user-defined course by simple placement of tangible objects and touch interaction. After starting the level editor, an initial simple course defined by a start and finishing line is shown on the surface. The start and finish lines can be repositioned and manipulated by touch interaction with the fingers. Each physical obstacle defines

Visual indicators help player drive through the waypoints.

Building parcours by placing waypoints with touch interaction.

Figure 4.25: An example course with physical obstacles and virtual oil slicks. Waypoints are placed through touch interaction.



a waypoint that needs to be passed by the cars. When placing an obstacle on the surface, the path gets adapted to include the new waypoint and by rotating the obstacle around its z-axis, the waypoint position around the obstacle changes depending on the orientation of the obstacle (cf. ► Figure 4.25). In this phase, oil slicks are added, repositioned, or removed by means of touch interaction.

4.3.5 User Feedback

A first version of TabletopCars was presented to eight computer science students in order to gather comments and early user feedback on the design and game concept. This first version already had all four games implemented and had to be played with the original remote controls. Extended functionality through the remote controller unit was not taken into account due to the early development status. The students learned the usage of the remote control within a few minutes and all students preferred the two-player mode due to the higher fun factor when playing against a human player. Overall, the students had never played such a game concept and were really keen on the idea to have active tangible cars enhanced through a digital environment. All players rated Car Crashing as the most preferred game with the highest fun factor because of the physical crashes of the cars and the act of chasing the opponent's car. The early user feedback showed that users quickly understood the interaction and the game concept with

Players preferred
two-player mode.

active tangible cars. The user's comments provided us with suggestions for improvements, for example, adapted placement of the point indicators nearby the cars. It turned out that players had difficulties with keeping their score in view if the score is displayed at a static position near the margins of the game area. Since their attention was focused on the cars, scoring events should be displayed nearby the cars.

Users quickly understood the game concept and preferred Car Crashing.

4.3.6 Embedded and Embodied Interaction

In order to enable embedded and embodied interaction, the Microsoft Kinect depth sensor was employed as an input device for interaction in TabletopCars. The depth sensor captured the whole body of players and tracked a rough skeleton model of them. Using the skeleton model, players can interact with their hand, their arm, or body gestures. For TabletopCars, the depth sensor was connected to the game by means of the FUBI¹⁹ (Full Body Interaction Framework) framework, which made use of the middleware OpenNI²⁰ for tracking the skeleton model. FUBI enabled easy recognition of static postures or a sequence of postures (as dynamic gestures) through providing simple XML declarations of the desired gestures.

Embedded and embodied interaction through a depth sensor.

We realized two simple sets of postures for an initial exploration of such kind of interaction with active tangible cars. The first set made use of the steering wheel metaphor known from real cars where both hands hold a virtual steering wheel in front of the player as shown in the skeleton model in ► Figure 4.26a. Turning the steering wheel to the left or to the right was directly mapped to turn commands to the controlled car. Pushing both hands forward or pulling them backward was mapped to forward and backward driving commands of the controlled car.

Steering wheel metaphor for interaction.

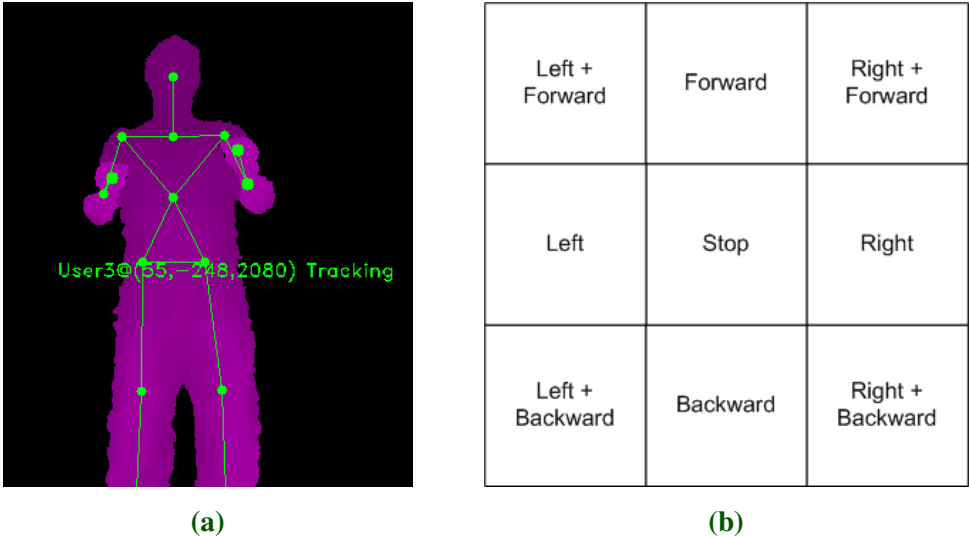
The second set of postures required only one hand to submit the commands and was based on a virtual "button-field" (cf. ► Figure 4.26b) spanned vertically in front of the user. Each button was triggered by moving one hand into the area of the button. Depending on whether the player was left-handed or right-handed, the stop button was adjusted to the left or the right at hip level in order to enable a comfortable initial position of the hands. Each button had a square form with the length of 25cm for a side.

Virtual button-field for interaction.

¹⁹<http://www.hcm-lab.de/fubi>

²⁰<http://openni.org>

Figure 4.26:
Kinect control
modes for
TabletopCars: (a)
Depth image of a
user in the
steering wheel
metaphor mode;
(b) Virtual
button-field mode
with the car
commands.



4.3.6.1 User Feedback

Preliminary study to
evaluate interaction
modes and gather
qualitative feedback.

For an initial exploration and to get user feedback on embedded and embodied interaction, we conducted a preliminary explorative study to investigate how users cope with this kind of interaction. We compared the presented set of postures with the original remote controls as the baseline in order to get comments on the interaction. Therefore, we recruited 20 volunteers (17 males and three females) aged from 22 to 34 with a mean age of 28. Each of them was right-handed and all of them were computer science students or researchers.

4.3.6.2 Procedure

In each session, two
participants played
Car Crashing. Both
interaction modes
were used to play by
every participant.

We chose Car Crashing for the exploration because of ratings from early user feedback, which stated Car Crashing as the easiest and most preferred game. Each session was carried out with two volunteers who played against each other at the same time. The players stood side by side at about 40cm distance to each other. Both players were positioned in front of the tabletop and the Kinect sensor. They were tracked by the Kinect sensor and had a clear view on the tabletop. Each volunteer was given an introduction to the game rules and all three interaction modes, that is, the original remote control, the Kinect steering wheel metaphor, and the Kinect button-field. The order of interaction modes was randomized for each pair of volunteers. After each introduction, the volunteers were given up to 5 minutes to get acquainted with the interaction mode. In this phase, players were instructed to try out each of the commands and to navigate their car for several circular laps over the surface one after another. As soon as each volunteer confirmed acquaintance with the interaction mode, the volunteers played one Car Crashing session. Finally, the volunteers were instructed

to fill out a questionnaire that asked them for comments on and preference of interaction modes.

4.3.7 Results and Discussion

The volunteers were asked to give a preference for an interaction mode where multiple choices were allowed. One of the volunteers preferred only the Kinect steering wheel metaphor and 19 of them preferred the original remote control. Of these 19, three also gave preference to the Kinect steering wheel and another one would also prefer the button-field. Since players reported in [Haller et al., 2010] about orientation difficulties with the cars, the questionnaire also asked for the best orientation awareness. The comprehension of the direction in which the car drives is important because left and right turning commands depend on the orientation of the car. Overall, we observed orientation difficulties with all three interaction modes because players corrected their steering commands every once in a while after they realized that the car moved into the wrong direction. Three of the volunteers stated the Kinect steering wheel metaphor as the only preference for orientation awareness, while 17 preferred the original remote control. Of the 17, four also stated the Kinect steering wheel metaphor as the preference and two other volunteers also gave the Kinect button-field the preference. The volunteers showed a preference for the original remote control, while 35% stated the Kinect steering wheel metaphor as a preference for the comprehension of orientation. The comments in the questionnaire and in talks afterwards revealed that most players had the feeling of more immediate and direct control with the original remote control compared to the additional latency induced by the Kinect sensing. 50% of the volunteers commented that they would prefer the Kinect steering wheel metaphor if the latency had been lower. They justified their choice with the novelty of the (Kinect) interaction technique. The latency from the Kinect sensing was not only a technical issue but also an issue of the amount of movements required to submit a command. Compared to a button press movement on the original remote control, the movement of the hands for the Kinect sensing showed a higher extent, which resulted in the feeling of additional latency. Further comments from players proposed that higher-level interaction techniques would be desired. For example, one hand to point at a location on the surface where the car shall drive to and the other hand for controlling the engine with finger postures.

Overall, all volunteers quickly understood the interaction modes and could cope with them. The preference for the original remote control stems from implementation issues such as latency and prior familiarity with the remote control. Furthermore, the comments showed that Kinect sensing would also be an accept-

Orientation difficulties.

Kinect steering wheel would be preferred if latency had been less.

able interaction mode if the interaction would be implemented more intuitive and easier to perform.

4.4 Chapter Summary

This chapter looked at the combination of multimodal tabletop interaction and tabletop tangible interaction. As we address only tabletop interaction related research, we omit the term tabletop to reduce redundancy in the writing unless we want to emphasize the relationship to tabletops. Tangible interaction is often argued as part of multimodal interaction. However, if we compare the quantities of research works for both, then the digital research libraries reveal much more research works that address tabletop tangible interaction than any other kinds of multimodal tabletop interaction. Note, that we consider only tabletop related modalities beyond touch. We, therefore, structured the design space parts in this chapter accordingly.

The first part of this chapter presented design possibilities for multimodal interaction in the form of a table (cf. ► Table 4.1) where each row details one kind of interaction with according perceivable modalities, interaction channels, medium, and appropriate codes and applications. Subsequently, we discussed each of the rows in the context of tabletop interaction in terms of limitations, potentials, enabling technologies, and concrete interaction techniques that have been investigated and evaluated in the research literature. The second part of this chapter presented a scheme to categorize tangible artifacts (cf. ► Figure 4.11) from a technological point of view, which further distinguishes the different forms of active tangibles. We then discussed the categorization scheme together with interaction techniques and implementation approaches proposed in the literature.

The remainder of the chapter presented our contribution to both fields and investigated the most advanced form of active tangibles (i.e., active actuated tangibles) together with embodied interaction techniques.

Each of the rows in ► Table 4.1 can be combined with each other to realize multimodal interaction and to design interaction techniques that go beyond sole touch. We showed and discussed combinations of the presented interaction kinds (with touch input) that have been investigated in the research literature. What we have not discussed are modality theories [Bernsen, 1993] in terms of how to combine multiple modalities (e.g., complementary, redundant, sequential, parallel [André et al., 2013]) or which combination would make sense. An appropriate consideration of this topic would go far beyond the scope of this dissertation. For this topic, we want to point at the dissertations of Tse [2007]

and Wasinger [2006]. However, the majority of works combine only one of the interaction kinds with touch input in a complementary manner.

The design space in this chapter (with its references and citations) mainly serves designers and developers to comprehend what modalities have been researched for tabletop interaction. In turn, what modalities (or combinations of modalities) have not been investigated provide fruitful directions for further research efforts.

Tabletop Centered Multi-Surface Interaction

TABLETOP centered multi-surface or cross-surface interaction constitutes the third part of the design space for tabletop interaction beyond touch. It is the most advanced subject and covers recent and ongoing rapid developments in today's device ecologies. More and more touch-enabled mobile devices (cf. ► Figure 5.1) and immobile devices (cf. ► Figure 5.4b) entered the consumer market since around 2007/2008 and the number of device types as well as the proliferation of devices is still increasing rapidly.

Trending and recent development.

A multitude of form factors has evolved since then, for example, smartwatches, smartphones, phablets, tablets, laptops, dual touchscreen laptops¹, all-in-one touchscreen desktop PCs, or touchscreen PCs that can also be operated as horizontal tabletops². Remarkably, the company Dell showed a commercial "smart desk", which integrates a large multi-touch tabletop into a productive working environment³.

Many form factors emerged and have become prevalent.

Nowadays, smartphones equipped with a capacitance-based touchscreen have become quite popular and evolved to the most personal computing device that we carry with us every day and everywhere. Similarly, tablet- and watch-sized touchscreen devices accompany us in daily life at home, at public places, and also at workplaces, for example, in offices or meeting rooms. With more and more such interactive surfaces around us, the number of environments, spaces,

What is tabletop centered multi-surface interaction?

¹ e.g., Acer Iconia Dual Screen

<http://www.acer.de/ac/de/DE/content/acer-iconia>

² e.g., Lenovo Ideacentre Horizon or Asus Transformer AiO P1801

https://www.asus.com/AllinOne-PCs/ASUS_Transformer_AiO_P1801/

³ <http://www.dell.com/learn/us/en/22/videos~en/documents~dell-smartdesk.aspx>



Figure 5.1: Examples of the variety of nowadays touch-enabled devices. From left to right: smartwatch, smartphone, phablet, mini-tablet, tablet, hybrid laptop.

and situations with a multitude of interactive surfaces grows as well. Sooner or later, this trend leads to the tempting desire to have and use applications that (1) span across multiple devices and (2) enable novel kinds of interactions with such applications in a coordinated, collaborative, or competitive manner.

Digital tabletops play a distinct role in such environments and device formations. The major difference to other device form factors, that is, their large surface and size, makes them static and immobile so that potential users have to gather around or stay nearby the tabletop. What looks like a drawback at the first glance actually turns out to offer many advantages and possibilities for applications in such environments if the benefits and the affordances of digital tabletops are taken into account. To name a few of them out of many, first of all, digital tabletops enable interaction tasks to be conducted seamlessly across multiple surfaces as digital tabletops provide the same basic interaction channel as mobile surfaces, namely touch.

Why digital tabletops
in multi-surface
environments?

Digital tabletops have much more computing power and resources than mobile surfaces, which allow performing computational intensive or extensive tasks – also on demand for mobile surfaces. In addition, tabletops do not suffer from limited battery capacity as they are constantly supplied with power, which makes them a reliable resource within the environment. In contrast, mobile surfaces may vanish unintentionally and suddenly without any notice caused by insufficient battery charge.

Examples for
advantages of digital
tabletops in
multi-surface
environments?

Furthermore, novel interaction techniques for such environments that function in a coordinated, collaborative, or competitive manner greatly benefits from a reliable entity within the environment, for example, to synchronize between devices, visualize common application state with more details, or to exploit the large surface for supporting human social behavior for territoriality [Scott et al.,

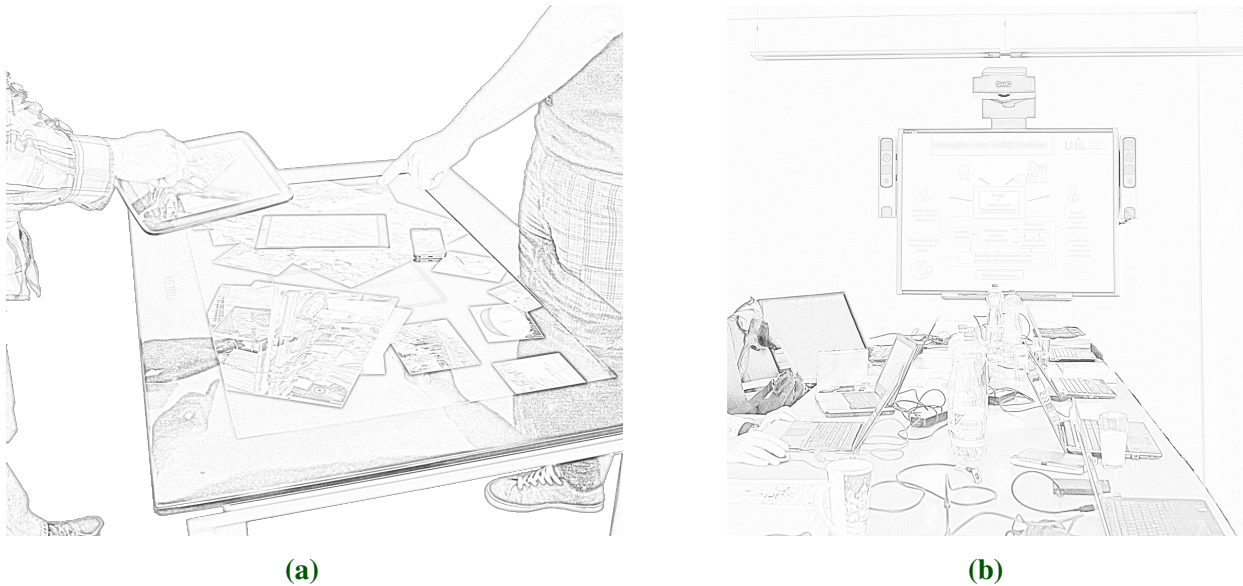


Figure 5.2: Example scenarios for multi-surface environment applications. (a) Photo sharing tabletop. (b) Meeting room environment.

2004]. Already those examples show the distinct features of digital tabletops in multi-surface environments that we systematically investigate and discuss in this chapter in order to provide means for designers and developers to make design decisions.

Before we continue to draw a detailed view on tabletop centered multi-surface interaction, let's portray a mental image of potential tabletop centered multi-surface environments by means of two simple example scenarios. Consider, for example, a large tabletop at a public place where people may casually engage in a theme-related photo sharing application (e.g., impressions of an ongoing festival) using their personal smartphone or tablet (cf. ► Figure 5.2a). People may grab a copy of certain photos on the tabletop that they like but only in exchange for relevant photos that they want to share and put on the public tabletop. This way, the tabletop operator as well as the users benefit from the multi-surface application. The tabletop operator receives a lot of relevant material that can be used for website content or promotional purposes and users get access to pictures or snapshots that other users are willing to share.

The second example targets occasions in meeting rooms at workplaces (e.g., similar to ► Figure 5.2b) where sharing and discussing digital documents in a meeting happen often. Currently, documents have to be exchanged in preparation for a meeting using conventional methods such as email transfer or USB sticks. A much more comfortable approach would be to use a shared multi-surface application that allows participants to distribute their documents with tablets or

Two simple examples to show tabletop centered multi-surface interactions.

Multi-surface environment in public places.

Multi-surface environment in meeting rooms.

smartphones that they carry with them anyway. In addition, a large projected surface, which is part of most meeting environments, could be included in the multi-surface application allowing participants to present the contents of their personal device directly on the large surface as well as control the presented content with their mobile device.

When we think of the initially described trending developments, occasions such as portrayed in the former examples are likely to become commonplace in the foreseeable future. The increasing number of recent research articles addressing applications and interactions within environments composed of mobile and large interactive surfaces endorses this trend (e.g., Cheung et al. [2014], McGrath et al. [2012], Cheng et al. [2012], Bachl et al. [2011], Gjerlufsen et al. [2011], Boring et al. [2010], Schmidt et al. [2009], Shen et al. [2006]). Today, however, enabling such applications and interactions requires considerable effort due to, for example, the range of nowadays heterogeneous device platforms. The fragmentation within each device platform (e.g., Google Android) makes this even more complicated. Furthermore, questions such as "how to divide control and content across devices?", "which interaction techniques might be appropriate for which task and device combination?", or "how to coordinate shared interactions?" are still subject to research.

There are many research works that address environments composed of multiple conventional displays, for example, conducted by the CSCW⁴ community. However, research for conventional displays construed for traditional mouse and keyboard interaction mostly cannot be directly transferred to environments with touch-enabled surfaces such as portrayed in the previous example scenarios. In those environments, users stand and walk around, interact through touch input on surfaces from a distance or remotely, or interact with collaborators through natural gestures. Hence, the usages of displays and style of interactions are quite different from that of users sitting in front of workstations with mice and keyboards [Beaudouin-Lafon and Mackay, 2003].

Overall, the rapid development of device ecologies with touch-enabled displays inevitably leads to more and more multi-surface environments to surface around us, which offer a huge potential for demanding multi-surface and/or cross-device applications. To date, development and design of such applications and interactions suffer from many open questions that need to be addressed by research. The increasing amount of recent MSE research works clearly confirms this.

In order to shape tabletop centered multi-surface interaction and environments and to foster research for this topic, this chapter establishes a basic understanding of multi-surface environments and informs about multi-surface tasks

Increasing interest of
MSE in research.

Bringing up
multi-surface
environments and
interactions is still
subject of research.

Many research works
for multiple
conventional displays
with keyboard and
mouse cannot be
transferred to
multi-surface
environments.

MSE research is
trending.

Structure of this
chapter.

⁴ Computer Supported Collaborative Work

and interactions within such environments. The remaining chapter first gives definitions for multi-display and (tabletop centered) multi-surface environments that serve to categorize existing research works. Afterwards, we present and discuss the design sub-space for tabletop centered multi-surface interaction that is supported by the results of an analysis of a thorough literature review of works according to the given definitions. Furthermore, we will show that an important requirement to realize research and implementations for such environments is an appropriate and flexible infrastructure. Therefore, we present the design, architecture, and implementation of a reference framework for this purpose called "Environs".

5.1 Definition of Multi-Surface and Multi-Display Environments

While the term multi-display environment (MDE) is almost self-explanatory due to its literal interpretation, it is not the case for the term multi-surface environment (MSE). Most people can easily imagine a room – or even happened to be in a room or an environment – with "multiple displays" installed, which is an appropriate mental image for MDEs. However, the literal meaning of a surface does not necessarily remind people of an interactive electronic surface in the first place. Everything around us has a surface, for instance, arbitrarily shaped physical objects, tools, tables, walls, floors, or doors. Thus, without further knowledge of the terminology, every room or environment could be considered a multi-surface environment.

Literal meanings of
MDE / MSE.

In existing research works, the most common usage of the terms refers to describing different compositions of multiple displays or interactive surfaces. Depending on the underlying research focus, the given descriptions further specify certain characteristics of such environments, however, without a common consensus on the requirements and capabilities of the involved devices or their environmental situatedness. Albeit, a clear understanding of those terms is important for a holistic view on tabletop interaction beyond touch. Therefore, we first create such an understanding before we continue to the substantial part of tabletop centered multi-surface interaction.

No clear common
consensus in research
works.

The definitions that we make in the following section are based on the results of a literature review of related works for which we favored recent works over older publications.

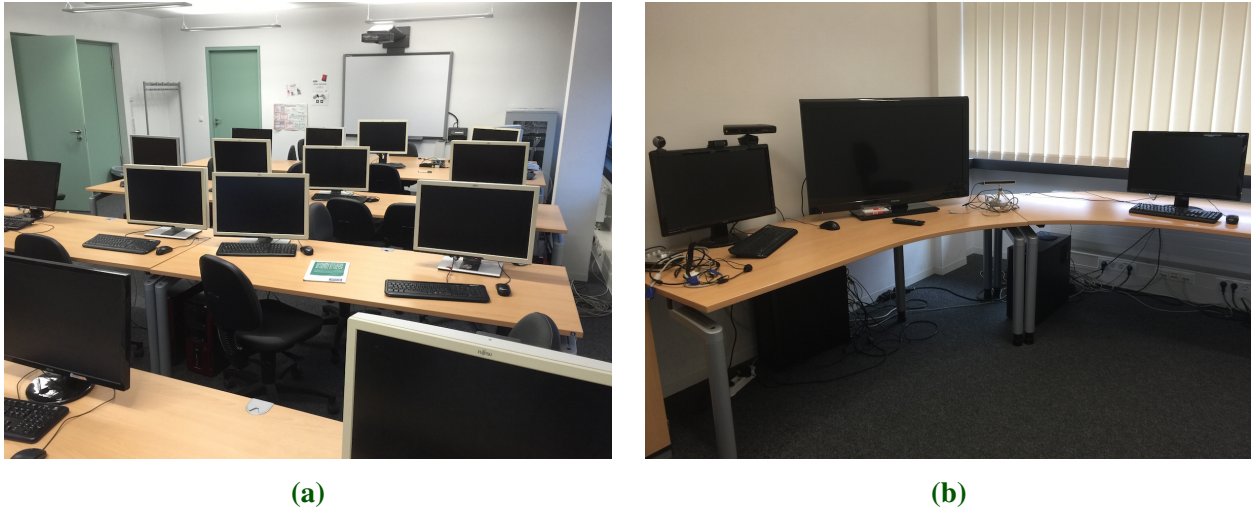


Figure 5.3: Examples for multi-display environments. (a) Conventional displays and projector in a classroom. (b) Multiple LCDs in a lab.

5.1.1 Multi-Display Environments (MDE)

MDE: Conventional displays. No direct-touch.

Multi-display environments literally denote physical places with more than one display, such as depicted in ► Figure 5.3. ► Figure 5.3a and 5.3b show multi-display environments composed of conventional displays, such as LCDs⁵ or projectors, which are usually not designed for direct-touch input, but rather for user input through mice, keyboards, or microphones. Applications may run as independent instances (e.g., a text editor) on each display or as shared and connected applications (e.g., collaborative web applications) distributed across displays. The environments in ► Figure 5.4a and 5.4b, however, exhibit significantly different characteristics due to their situatedness, affordances, and form factors. Passersby would not look for mice and keyboards in the first place in order to use applications provided by the environment's displays. Instead, rather speech input, body gestures, or personal smartphones would be assumed in ► Figure 5.4a. In ► Figure 5.4b, touch input on the tabletop surfaces or tablets would be expected due to nowadays prevalent experience with touch-enabled devices.

MSE: Displays allowing direct-touch.

Step by step beyond multiple conventional displays.

Step by step beyond mice and keyboards.

Most related publications.

In the following, we cite the most related publications out of all related works from the literature review. Citations are further presented with the statements that contributed to our definitions in order to lay the foundation for our definitions. Furthermore, we narrow the discussion of existing works down to those that are not solely based on mouse and keyboard input but also include or focus on touch input.

⁵ Liquid Crystal Display



Figure 5.4: Examples of multi-display / multi-surface environments. (a) Wall-displays in a corridor. (b) Multiple tablespots, tablets, and a notebook display.

Of the research works that explicitly use the term MDE, most of the presented device ecologies include large static displays combined with small mobile displays. For example, Biehl et al. [2008] described an MDE as a composition of "*co-located personal (e.g., laptops) and shared devices (e.g., large displays)*". The description of Forlines and Lilien [2008] given for MDE referred to concrete form factors and capabilities: "*a tabletop display capable of sensing touch-input, two large vertical displays, and a Tablet-PC*". Bachl et al. [2011] made "*additional input devices with displays*", such as "*mobile phones, laptop computers, pen-based tablet computers*" a requirement for their MDE. Seyed et al. [2012] stated a definition for MDEs, which also characterized the interaction within such environments: "*a system where interaction is divided over several displays, such as digital tablespots, wall displays and personal devices like tablets or mobile phones*".

Explicit usage of the term MDE.

What kinds of devices belong to MDEs?

Do MDEs require more than different device types?

Sometimes both terms (MDE and MSE) were also used interchangeable. For instance, Abad et al. [2014] and Chokshi et al. [2014] cited the MDE definition from Seyed et al. [2012] and reused it without modifications to define MSEs. Nacenta [2009] also made no difference between MDEs and MSEs. However, Nacenta specified the displays for such MDEs as "*an array of light elements that are arranged in a continuous and regular way along the same surface in order to show dynamic output from a computer*". Hence, Nacenta's definition clarifies the usage of display and excludes for example static (non-electronic) displays such as large paper poster ads on walls.

Usage of MSE and MDE is often interchangeable.

5.1.2 Multi-Surface Environments (MSE)

Early research work emphasized direct interaction with displays.

So, how are MSEs distinguished from MDEs in the existing literature? Early works that attempted to describe distinguishing characteristics of multi-surface environments – vs. multi-display environments – emerged around 2006. At that time, Shen et al. [2006] explicitly distinguished between MDEs and MSEs by the nature of additional input through the display: *"By using the term multi-surface, instead of multi-display, we emphasize the nature of many of today's interactive walls, tables, Tablet PCs, desktop displays, laptops and PDAs that often can be interacted upon in addition to be merely the visual display"*.

Later research works followed this perspective and put further emphasis on the interaction aspect. For example, Klokmoose and Beaudouin-Lafon [2009] *"specifically address multi-surface interaction, i.e., interaction spanning the surfaces of multiple devices"*. A similar description was given by Gjerlufsen et al. [2011] who defined MSEs as *"ubiquitous computing environments where interaction spans multiple input and output devices and can be performed by several users simultaneously"*.

Similar to the definition approach in ► Subsection 4.1.2, the following definitions are consecutively constructed. We first define multi-display environments and give on this basis a definition for multi-surface environments and tabletop centered multi-surface environments.

5.1.3 Definitions for MDE and MSE

Definition of MDE.

This dissertation considers MDE as a generic term for environments that include all kinds of workspaces with multiple displays similar to the view of Nacenta [2009].

Definition 8. *Multi-Display Environment.*

*A multi-display environment (MDE) is an interactive computing environment composed of one or more computing systems and with two or more large electronic displays that **form a logically connected workspace**.*

What's not an MDE?

According to ► Definition 8, multiple displays that are solely connected through network links cannot be considered as an MDE. They further have to implement applications that provide a logically connected workspace in order

to be an MDE, for example, by synchronized views or interactions. Based on ► Definition 8, we define MSEs as a subset of MDEs and confine MSEs through the understanding of Shen et al. [2006] who emphasized interactions within MSEs. The following requirements must be met for an MDE to be an MSE.

Definition of MSE.

Definition 9. *Multi-Surface Environment.*

1. *A multi-surface environment (MSE) must meet the requirements of a multi-display environment according to ► Definition 8.*
2. *At least one of the large displays has to be an interactive surface. Interactive surfaces are electronic (or projected) displays or have electronic displays embedded, which provide touch input to interact with the multi-surface environment, for example, digital tabletops, tablets, or smartphones.*
3. *In addition to that, the logically connected workspace of an MSE has to span a common synchronized interaction space.*

Note that the interactive surfaces of an MSE are not limited to consumer devices (e.g., tablet, smartphone, digital tabletop) but may also include projected interactive surfaces, such as arm projections as demonstrated by Adachi et al. [2013]. Furthermore, devices of an MSE do not need to be situated in the same physical environment. Usually, most of them are situated in the same physical environment but may also be situated in multiple different physical environments, such as different locations on board of a ship [Domova et al., 2013].

MSEs are not limited to touch displays or one physical location.

5.1.4 Tabletop Centered MSE

We further constrain MSEs to a subset that has at least one digital tabletop as the central interactive surface for interaction to meet the scope of this thesis. If we mention MSE, then we refer to tabletop centered MSEs. In particular, we consider tabletop interaction beyond touch in tabletop centered multi-surface environments.

Tabletop centered MSE.

5.1.5 Alternative Terms

Table centric
interactive Spaces.

Active Spaces.

Interactive
Workspaces.

MSE terms taken over
from ubiquitous
computing.

Besides the use of the terms MDE and MSE, several different expressions and paraphrases have been used for such interactive environments. The terms that are most related to our definition of MSEs are the "table-centric interactive spaces" or "table-centric multi-surface environments" of Wigdor et al. [2006]. Jetter [2013] named such interactive spaces as "ZOIL" for zoom-able object-oriented information landscapes according to the object-oriented design paradigm of their system. ZOIL applications were based on zoom and pan activities for navigation within information landscapes. Román et al. [2002] chose a likewise general term and denoted such environments as "Active Spaces", which is quite similar to the term of Ponnekanti et al. [2003] who used the simple term "Interactive Workspaces". Rekimoto and Saitoh [1999] augmented working environments with projected surfaces and called them "Hybrid Computing Environments".

Many terms were borrowed from the ubiquitous computing community such as "Roomware Environments" of Tandler [2000] or "Active Spaces" of Ponnekanti et al. [2003]. Although MSEs have many research concerns with ubiquitous computing environments in common, MSEs have a strong focus on interaction with and through interactive surfaces. Ubiquitous computing environments, however, embraces all kinds of computing devices that vanish into the background of that environments.

5.2 Usages in Tabletop Centered Multi-Surface Environments

The research contributions, insights, and statistics discussed in this chapter are grounded on a thorough review and analysis of research literature that are relevant to tabletop centered multi-surface environments, interactions, and tasks. Moreover, the results of the literature review also serve to shed light on the current research state (till mid-2015) and to establish a design space for tabletop centered multi-surface interaction.

5.2.1 Literature Review

Since interaction with and across multiple displays has long been the subject of interest in the HCI community, there is a large body of research works related to multi-display and multi-surface environments as well as enabling technologies. As we also have discussed so far, many of the research works are not relevant for

nowadays multi-surface environments defined in ► Definition 9. In particular, the wide proliferation of mobile devices or direct-touch as the dominant input modality for those devices were not envisioned in older publications. To sort out not relevant works, we conducted a literature review for works that meet multi-surface environments according to our definition of tabletop centered MSEs.

Literature database
included research
according to
► Definition 9.

Out of the many papers that were collected, 89 relevant papers were identified, which are listed in ► Appendix A.4 with authors and title sorted by reference indexes. ► Appendix A.5 lists the same relevant papers sorted by author names. Papers were collected by means of a systematic process in which the first step was a keyword search for "multi display", "multi surface", "multi display environment", and "multi surface environment" (including variations such as "multi-surface", "multisurface", or "multiple surface") through the electronic databases ACM Digital Library, Springer-Link, IEEE Xplore, and Google Scholar. All keyword combinations were required to build a thorough database due to the following reasons. The last two keywords were required as topics covering "multi-touch" often occurred in combination with "surface" or "display". Hence, search results were "flooded" with multi-touch related hits, but not necessarily related to multi-surface environments. However, the results of a search for only "multi surface/display environments" would miss many relevant works that use synonyms for multi-surface or environments, such as active/interactive spaces, workspaces, smart environments, or collaborative computing, see the alternative terms in ► Subsection 5.1.5.

Paper collection
process.

After sorting out irrelevant papers, the conference proceedings in which the relevant papers were published were collected and manually searched through in order to find more relevant papers, for example, CHI, ITS, EICS, IUI, UIST, AVI, PerDis, or UbiComp. To find relevant papers on the basis of author citations, forward snowballing [Greenhalgh et al., 2005] on the relevant papers was performed. Literature references of the relevant papers were followed and collected until no more relevant cited papers were found. Relevant full papers then went through backward snowballing [Greenhalgh et al., 2005] to find literature that cited the identified relevant full papers. Backward snowballing, however, resulted in only a few additional relevant papers. The majority of relevant papers were found through the previous approaches.

Conference
proceedings.

Forward snowballing.

Backward
snowballing.

In order to show an objective overview of the research landscape, we excluded our own publications from the analysis and statistics. Instead, the analysis and statistics serve to justify our efforts and contributions.

Relevance of each paper was judged primarily based on the investigated device types and configurations (according to ► Definition 9). For example, if no interactive touch-enabled tabletop was part of the device ecology or if the paper did not explicitly target interactive tabletops, then the paper was considered not

Judgment of
relevance.

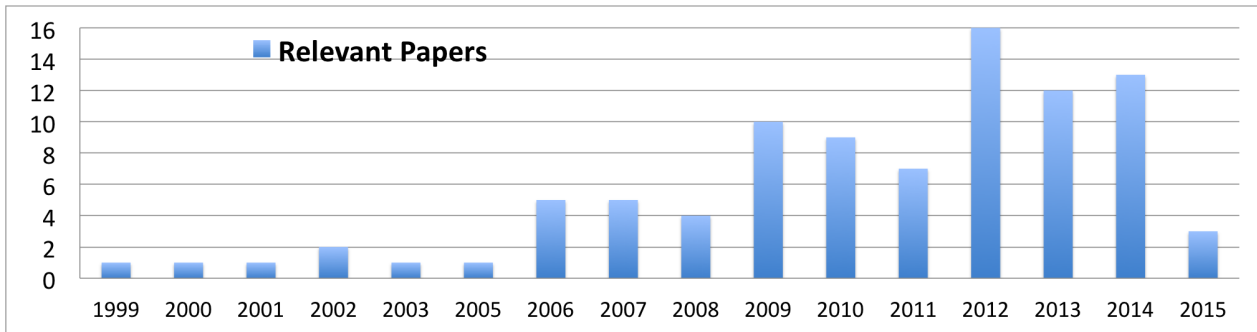


Figure 5.5: Number of relevant MSE papers distributed over publication years.

relevant. Research works that do not target interactive tabletops usually lack of considering certain affordances of tabletops, such as those originating from the large horizontal surface or the influence on collaboration in comparison to vertical displays. The majority of the papers were full papers (51) and short papers (5). Ten extended abstract papers were included due to their high relevance for multi-surface environments. The histogram given in ► Figure 5.5 shows the distribution of research works over the years since 1999. The interest within the research community began increasing in 2006, which correlates with the availability of cost-effective FTIR tabletops around 2005. Furthermore, ► Figure 5.5 clearly indicates an upward tendency until 2014, which can be explained through the increasing proliferation of mobile interactive surfaces.

5.2.2 MSE Design Space

Only a few design spaces for multi-surface interaction have been proposed in the past, whereof each address different underlying design questions and perspectives. We will briefly discuss the relevant works to establish an overview of previous work.

Early work of Shen et al. [2006] proposed three modes of multi-surface visualization and interaction to characterize shared content, visualization, and UI manipulation within MSEs. Depending on the connectedness and interrelationship of the distribution across surfaces, interaction and visualization are attributed to either "Independent", "Reflective" (distributed input and output are tightly coupled), or "Coordinated" (distributed input and output are interdependent but not required to be identical).

Later, Nacenta et al. [2009] presented a design space intended for cross-display object movement (CDOM), in which interaction techniques are classified to the three layers "Reference" (spatial, non-spatial), "Configuration" (planar, perspective, literal), and "Control" (open-loop, intermittent, closed-loop).

Design spaces in the literature.

Cross-display object movement.

Each layer suits well to characterize CDOM techniques but falls short in mapping MSE interaction and tasks in general.

The remaining works have in common that they consider input type (using the device, through surface), or input and output device as the dimensions of the design space. They differ in the underlying design questions that the proposed design space aims to answer. For example, Ajaj et al. [2009] oppose input devices (tabletop, mouse) that are available in the real world to the tasks performed (select, translate, scale, rotate) in the virtual world and connect both by means of a range of interaction techniques as an intermediate layer. Their design space aims at studies to compare mouse and tabletop techniques to manipulate 2D/3D perspective views for which the intermediate layer arrange available design choices. Other works additionally include output and visualization to this schema, for example, Schmidt et al. [2012] (surface, mobile) and Spindler et al. [2014] (global, local, tangible, active views). Wagner et al. [2013] add a body-centric perspective to the schema and consider input and visual output to be relative to the body or fixed in the world in combination with the body involvement of users, that is, which part of the body is actually involved in the interaction.

In comparison to previous works, our design space aims to provide an overview of interaction techniques and enabling technologies following the structure of the previous chapters of this dissertation. The intention of this design space is to provide assistance for design considerations when designing tabletop centered MSE interaction beyond touch. Consequently, the analysis of the relevant literature described in ► Subsection 5.2.1 was driven by the following research questions:

- Q1.** What interaction techniques have been proposed and investigated for MSEs?
- Q2.** What tasks and applications have been investigated for MSEs?
- Q3.** What enabling technologies have been proposed?
- Q4.** What are the logical connections and relations between the interactive surfaces in MSEs?
- Q5.** What base technologies related to interaction and technologies have been investigated in MSEs?

Input, output, tasks,
and interaction as the
glue.

Body-centric
perspective.

MSE interaction
beyond touch.

Research questions
for the literature
review.

Table 5.1: Classification of MSE interactions

		Subcategory	Description
Interaction	Input	Touch-only gestures (26,6%) [P: 5, 6, 10, 13, 16, 24, 25, 36, 37, 41, 42, 45, 49, 55, 56, 60, 63, 68, 73, 81, 82, 87, 90, 91]	Gestures, such as scale, translate, pinch, or flick, that are performed on a surface with only touch.
		Mobile device gestures (23,4%) [P: 4, 5, 6, 20, 21, 22, 26, 31, 42, 45, 54, 57, 58, 60, 62, 66, 67, 70, 72, 74, 78, 84, 87]	Gestures, such as flick, pour, shake, fold, or bump, that are conducted using spatial movements of a mobile device.
		Pen (17,2%) [P: 10, 11, 14, 15, 39, 44, 49, 57, 63, 64, 70, 80]	Interaction that involves pens to draw or touch on surfaces.
		Tangibles (3,1%) [P: 6, 11, 70]	Use of physical artifacts as tangible objects.
		QR-Code / NFC (3,1%) [P: 20, 23, 43, 81]	Interaction that involves QR-Codes or NFC.
	Task	Navigate (37,5%) [P: 5, 6, 10, 11, 12, 15, 16, 17, 20, 21, 23, 25, 36, 38, 41, 42, 44, 47, 48, 55, 56, 57, 58, 63, 65, 67, 68, 70, 71, 72, 75, 79, 80, 81, 84, 85, 88, 90, 91]	Navigation through virtual workspaces or menus; Browsing through items; Happens on local or remote surfaces.
		CDOM (34,4%) [P: 5, 6, 8, 15, 16, 19, 20, 21, 24, 26, 27, 28, 29, 37, 42, 45, 47, 51, 54, 55, 60, 62, 67, 73, 74, 78, 81, 82, 87, 90]	Cross-device object movement includes copy, paste, move, or transfer data between devices.
		Pointing (28,1%) [P: 4, 5, 11, 16, 18, 37, 39, 40, 41, 42, 50, 51, 53, 55, 57, 62, 64, 69, 70, 71, 73, 74, 82, 87, 89, 90, 91]	Point and selection tasks performed on local or remote surfaces or using spatial gestures.
		Annotate (6,3%) [P: 6, 44, 49, 58, 70, 72, 80, 85]	Input and adhere text to objects.
	Channel	Separate views (25,0%) [P: 3, 12, 13, 19, 20, 23, 38, 41, 43, 51, 53, 54, 56, 57, 58, 67, 68, 70, 71, 72, 73, 74, 85, 86, 90]	Use of multiple surfaces for separate, alternative or additional views.
		Interactive portals views (23,4%) [P: 8, 13, 14, 15, 24, 25, 34, 42, 44, 47, 48, 53, 54, 55, 56, 57, 69, 71, 90]	Real time video based remote interaction with remote content. Alternative terms: Porthole (56), Lens (48, 56), Fovea-Tablet (14, 15, 44), Screen forwarding (53).
		Synchronized views (14,1%) [P: 10, 12, 18, 26, 36, 38, 41, 46, 59, 65, 70, 71, 75, 76, 77, 79, 80, 84]	Keep content or views across multiple surfaces in sync.
		Camera see-through views (1,6%) [P: 42]	Using mobile devices as touch-enabled see-through displays.
	Base Technologies	Frameworks (23,4%) [P: 5, 6, 11, 15, 16, 21, 30, 31, 32, 33, 34, 35, 53, 59, 61, 65, 70, 71, 76, 77, 86, 89]	Either presenting a framework or work that relates to an MSE framework.
		User/Device location, perspective (17,2%) [P: 4, 5, 6, 8, 21, 30, 31, 34, 40, 41, 47, 64, 69, 74, 75, 81, 82, 83, 84, 87, 90, 91]	Tracking of user's location or user's viewing perspective depending on his location or the location of his head.
		Personalization, Authentication (10,9%) [P: 16, 19, 25, 43, 54, 60, 62, 73, 85, 88]	Provide content depending on which user performs the interaction or which user requests the data.
		Privacy, Security (7,8%) [P: 6, 36, 38, 54, 62, 67, 71, 72, 73, 75, 78, 87, 90]	Approaches to keep private data only accessible to the belonging user.

► Table 5.1 lists the results of our analysis, which represent the design space for tabletop centered MSE interaction beyond touch. The table is divided into four main categories (second column) with each having several subcategories (third column) and brief indicative descriptions (fourth column). Categories one and two (Input, Task) address the questions ► Q1 and ► Q2 and denominate common dimensions that are also used by previous design spaces to describe means and aims of interaction techniques, for example, in [Ajaj et al., 2009, Schmidt et al., 2012, Spindler et al., 2014]. The remaining categories address questions ► Q3, ► Q4 (Channel) and ► Q5 (Base technologies) and show interaction related possibilities and challenges as well as interaction related base technologies that the majority of the research works have in common. Those main categories form a four-dimensional design space to discuss and judge on criteria with regard to concrete interaction techniques.

Structure of Table 5.1.
Four categories.

Input, Task.

Channel, Base
technologies.

LIMITATIONS AND VALIDITY. It is important to note that the subcategories of the main dimensions do not represent an exhaustive list but were developed by means of the analysis of relevant literature. Thus, they do reflect the state-of-the-art in MSE research interests and may be extended in future as a result of future research and advances on this topic.

5.2.2.1 Input

The category *Input* addresses research question ► Q1 and overviews input mechanism and interaction channels that have been proposed for interaction techniques in MSEs. Instead of enumerating every device or gesture that was used in the existing literature, we grouped them into a set of inclusive and meaningful subcategories and counted the frequency of use throughout the relevant papers. As this dissertation is about tabletop interaction beyond touch, sole touch input (e.g., touch-down, touch-up) is omitted in the MSE design space (► Table 5.1) in order to portray a focused picture of the potential for interaction beyond touch.

Input beyond touch.

The majority of research works investigated or implemented gestural interaction. Of these works, touch-only gestures were used slightly more often than gestures with dedicated input devices. Touch-only gestures were performed either to manipulate digital objects or to navigate through digital workspaces (2D/3D). Gestures for object manipulation include, for example, multi-touch or bi-manual touch gestures to scale, pinch, rotate, flick, or move digital objects on a surface. To navigate through workspaces provided by surfaces, gestures such as pan or slide were mainly used.

Touch-only gestures.

Gestures performed with mobile devices (e.g., smartphone, tablet) were used almost as often as touch-only gestures. The investigated interaction techniques treat mobile devices as spatial controllers that enable, for example, pour ges-

Mobile device
gestures.

Recognition of device
gestures.

tures (e.g., ► P6, ► P21), flick gestures (e.g., ► P26, ► P45) or bump gestures (e.g., ► P45, ► P54, ► P60, ► P62, ► P66). Recognition of such gestures made use of two kinds of sensors. The first and most often used kind were sensors embedded into mobile devices such as accelerometer or gyroscope (e.g., ► P54, ► P62). Those sensors enable quite fast and accurate tracking of mobile devices' spatial movements or state changes thereby enable responsive interaction techniques. The second kind of sensors was situated in the environment to observe users and users' devices such as elaborate motion tracking systems in ► P11 or ► P22 (Vicon). Those systems are usually expensive and tightly tied/calibrated to the target environment. As an alternative to those systems, research works investigated promising approaches using low-cost depth cameras, for example, in ► P30 (Microsoft Kinect), which, however, induce additional latency into the whole interaction and in turn negatively impact the user experience. However, with advances and newer generation of depth cameras, this approach will also become much more accepted by users.

Pens.

The third often addressed input subcategory is characterized by interaction with pens. Pens are more advantageous than touch gestures in terms of precision and accuracy at pixel level, which is beneficial, for example, to input written text or draw sketches. Examples that require such precisions were annotation (► P44) or precise pointing (► P39).

Tangibles.

Finally, the fourth and least often used input subcategory includes interaction with graspables, physical artifacts, and tangible objects (► P6, ► P11). Even though only a few works employed this input category, it is a promising approach for MSE interaction by reason of the potentials discussed in ► Section 4.2. For example, Chokshi et al. [2014] exploited the reduced learning curve of tangibles in an emergency response planning application in which every second counts. Gjerlufsen et al. [2011] integrated a tangible representation of a human brain to allow for intuitive access to 6-DOF navigation tasks.

5.2.2.2 Tasks

General types of
tasks.

The second main category of ► Table 5.1 summarizes actions and effects caused by interaction techniques into the descriptive subcategories Navigate, CDOM (Cross-Device Object Movement), Pointing, and Annotate. More general tasks that were found in the relevant literature were decomposed into the given subcategories. For example, sorting tasks were decomposed to Navigate and CDOM if both were involved in the particular task. More concrete tasks were assigned to one of the general task descriptions. For example, manipulation of objects may be attributed to CDOM or Navigation depending on the aim of the task.

The most often referred tasks belong to the subcategories Navigate and CDOM. Navigation includes tasks in which digital objects, physical objects (e.g., smartphones), or digital workspaces are manipulated with the aim to navigate, (re-) orientate, or search, for example, browsing through a spatial workspace to find certain regions (► P48, ► P57). CDOM includes tasks that involve the transfer of digital content across surfaces, in particular, copying, pasting, or moving of digital objects from one surface to another surface. All CDOM tasks have the transfer of content in common. Depending on the tasks, additional operations may occur before and after the content transfer. For example, a copy operation leaves the content on the source device untouched, whereas a move operation deletes the original data from the source device and creates the transferred data on the target device or in a temporary clipboard. CDOM was extensively investigated in the Ph.D. dissertation of Nacenta [2009] and represents a default task due to its strong affordance when working in environments with multiple source and target surfaces.

 Navigation

 CDOM –
Cross-Device Object
Movement

The third often implemented task realized pointing at distant objects or locations either through spatial gestures (with mobile devices), body gestures, or gestures on a surface (e.g., by means of replicated views from distant surfaces). The spatial arrangements of devices in tabletop centered MSEs virtually invite users to select distant targets (on distance tabletop surfaces) by means of interaction techniques using mobile surface. Besides using mobile surfaces, research works also addressed this task by means of techniques performed on a tabletop surface through multi-touch techniques (e.g., ► P37).

 Distant pointing.

The least often occurred task involved annotation of objects or locations, which is most often conducted through pens. Mobile surface devices that support pen input are particularly suitable in MSEs to serve as a keyboard replacement as annotation tasks require only small portions of text, for example, for keywords, references, names, or notes.

 Annotation.

5.2.2.3 Channels

The third main category refers to interaction channels as defined for multimodal tabletop interaction in ► Chapter 4 and addresses research question ► Q1, ► Q2 and ► Q4. Throughout the relevant literature, we could identify three frequently used channels, which have relevance to the modes of interaction outlined by Shen et al. [2006] (i.e., Independent, Reflective, Coordinated). In addition to them, we discuss a fourth quite challenging channel that exploits the highly powerful camera technologies of nowadays mobile devices.

The first three channels in ► Table 5.1 represent the most frequently used input/output combinations to interact with a distributed workspace and to visualize

 Separate Views.

the workspace state. In the "Separate Views" subcategory, at least two different surfaces serve as separate and different views of the workspace. Large tabletop surfaces give access to the workspace and serve as main views of the workspace whereas mobile devices provide additional utilities such as a data storage or tool-palette (e.g., ► P24, ► P54), a menu (e.g., forearm-menu ► P3), or a personal game view (e.g., ► P67). The visual outputs of those views are independent of each other in the sense that they show and represent different content of the workspace or content (e.g., menus) that is related to the workspace.

 Synchronized Views.

In the "Synchronized Views" subcategory, at least two separated surfaces show interdependent views of the workspace. The visual representation of synchronized views relates to the same or interdependent (workspace) content or state. If the content or state changes, then the changes are reflected through visual means on the synchronized views if appropriate. For example, the VisPorter system in ► P26 makes use of tablets to present a concept map while tabletops and wall displays synthesize the concept map with details of the concepts. Changes on the concept map are synchronized to each visualization whereas changes on the details are (usually) only synchronized to larger surfaces.

 Interactive Portal Views.

The most salient channel is represented through "Interactive Portal Views", where at least two different surfaces show the same visual representation of the workspace through views. The visual representations of the views do not necessarily exhibit the same scale and location within the workspace. Changes of the content in the workspace are instantly reflected on every portal view if appropriate. Such portals can be realized, for example, as real-time video replications of the workspace (e.g., ► P55), real-time video that extends the workspace (e.g., ► P8), or synchronized views showing exactly the same visual representation on both (e.g., ► P56). Together with touch input performed on the views, which modify the global workspace state, those channels are called "Interactive Portals". Within the literature, such portals are also called porthole (► P56), lens (► P48, ► P56), fovea-tablet (► P14, ► P15, ► P44), or screen forwarding/sharing (► P53, ► P71). Interactive portals enable a variety of demanding novel applications, such as real-time semantic overlays of workspace areas [Dang et al., 2015], command and control on board of ships (► P8), or collaboration with field-workers in industrial settings [Domova et al., 2014]. It is also suitable for easy implementation of otherwise intricate to realize applications such as (mobile, high resolution) focus + (large, low resolution) context applications (► P56).

 Camera see-through views.

The last channel realizes camera-based interaction by exploiting mobile devices and its cameras as touch-enabled see-through views (► P42). Such camera views make use of video cameras to capture the (tabletop) display views (in the environment) and track its positions by means of (form/shape) knowledge of the displays. Touch interaction on mobile devices happens as if they occur on target

surfaces (and views). Even though only one research work elaborated on camera see-through views in a tabletop centered MSE context, this interaction channel is listed in ► Table 5.1 due to two reasons. It represents a significant different logical view connectedness in comparison to the other interaction channels and provides promising interaction techniques with mobile devices in tabletop centered MSEs (e.g., [Boring et al., 2010]). Furthermore, as camera technologies in mobile devices are expected to advance with each new generation of mobile devices, this development will also foster camera see-through views and its potential for tabletop centered MSE.

5.2.2.4 Base Technologies

The last category includes base technologies that were frequently addressed in many of the relevant papers. Hence, by implication, the research works ascribe those topics to be of high importance. Therefore, they are highly recommended to be addressed in design considerations for tabletop centered MSEs. ► Figure

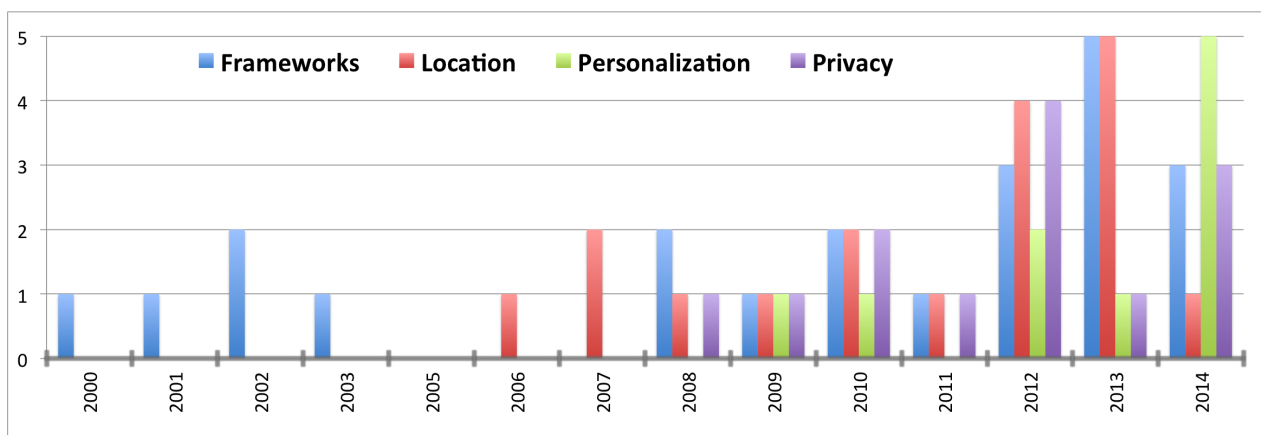


Figure 5.6: Number of papers distributed over publication years for the base technologies: MSE framework, Location / Perspective, Personalization, Privacy/Security.

5.6 sketches the frequency of the research works dedicated to the base technologies distributed over publication years. The most often addressed base technology aimed at the technical aspect of enabling technologies in the form or frameworks, toolkits, or middlewares as listed in ► Table 5.1. In this work, we use the term framework in order to follow the naming convention of recent publications (e.g., ► P30, ► P31, ► P53, ► P65).

Early works rather addressed ubiquitous environments instead of multi-surface environments, for example, Roomware Environments (► P35), Active Spaces (► P33), or Interactive Workspaces (► P32). However, they all embrace interactive tabletops as part of the device ecology. At that time, mobile surfaces consisted of pen-driven laptops or touch-enabled laptops, which are quite

Frameworks.

different in terms of software and hardware in comparison to the capabilities of nowadays smartphones or tablets. Hence, the proposed toolkits made use of technologies of that time (e.g., CORBA in ► P33) and are hardly available and adaptable to nowadays software and hardware technologies of mobile surfaces. With the rapid proliferation of mobile touch-enabled devices, the number of research works investigating frameworks as enabling technologies (that include mobile surfaces) also increased. Frameworks belong to the essential enabling technologies as they provide basic functionality to establish MSEs. Hence, MSE framework concepts and architectures will be discussed more detailed in the next ► Section 5.3.

User location and
perspective.

The second often addressed base technologies consider the location of users, users' body pose/position, or the location/pose of mobile devices as part of the spatial arrangement within MSEs. While interactive tabletops or large vertical displays are usually immobile and statically positioned in the environment, the location and pose of mobile surfaces and users are not fixed and may change regularly, which in turn have implications for the design of appropriate input and output. In terms of input aspects, the proximity of devices and users, as well as their pose in relation to each other, have a strong influence on appropriate interaction techniques. For example, a flick gesture would be suitable for CDOM operations between distant surfaces whereas a pick-and-drop gesture might be more comfortable between surfaces in close proximity. In terms of output aspects, the viewing perspective of a user changes with head location and pose, which needs to be considered in the design of UIs as well as for content distribution across surfaces. For example, a user sitting at a large tabletop may not have an undistorted view on the whole surface. Here, distorted content can be made accessible through views on the mobile device of the user. Recognition of spatial arrangements (of users and devices) in an MSE was achieved by means of elaborate (but expensive) motion tracking systems (e.g., Vicon in ► P11 or ► P22) or cost-effective (but less precise) consumer depth cameras (e.g., Microsoft Kinect in ► P30).

Personalization,
Authentication.

The remaining two base technologies, that is, personalization and authentication technology, concern MSE scenarios that include more than one user. In such scenarios, the large surfaces become shared workspaces and mobile devices are used as private/personal surfaces. Both base technologies have a strong relationship to each other in the sense that they consider input and output mechanisms, which either adapt content according to user preferences or constrain access to private (or user-related) content in the distributed workspace. Personalization enables user-dependent views, which might be adapted depending on user preferences, for example, to include user avatars or user-defined gesture sets. Authentication enables users to identify themselves to the MSE application. Once authenticated, MSE applications might apply personalization or give access to

private data. The awareness of privacy and security of data has increased noticeably during recent years caused by the general trend of today's internet services to collect more and more user-related data. In MSEs, privacy and security concerns become more obvious when mobile devices are used as private views. In order to keep those views private, appropriate interaction techniques, as well as suitable output visualizations, are required, which may be different if mobile surfaces can be used as public views as well.

Privacy, Security.

5.3 Enabling Technologies

Establishing multi-surface environments and enabling user interaction within such environments as discussed in the previous section require two distinguishable and essential components, which have to work seamlessly together.

Components to enable MSEs:

1. Sensor and display technologies, as well as devices, that provide interactive surfaces and means to enable user interaction.
2. Infrastructure technologies to establish a distributed workspace and to enable synchronized interaction techniques as well as visualization across the interactive surfaces within an MSE.

Both components are determinant for viable applications and interaction techniques for MSEs. The better one component works with another, the better the user experience that can be achieved. They "touch" every category and subcategory of the design space discussed in ► Section 5.2. Therefore, they are considered as key components and critical for the success of MSEs.

5.3.1 Sensor and Display Technologies

Sensor and display technologies employed in multi-surface environments are based on the technologies that we already have discussed in detail in ► Chapter 3 (Touch Interaction) and ► Chapter 4 (Multimodal and Tangible Interaction). Hence, we will briefly discuss them in the context of multi-surface environments.

Nowadays prevalent mobile devices (e.g., smartphones, phablets, tablets, or smartwatches) predominantly make use of capacitance sensing to realize touch or pen interactions on the display surface. Wall displays usually make use of capacitance sensing if they are capable of input mechanisms (e.g., Microsoft Surface

Touch technologies.

Hub⁶). Very large wall displays or interactive walls usually employ one of the vision-based sensing technologies in ► Table 3.1 as those technologies provide easier and cost-effective scalability of the interactive surface's dimensions. The most variable usage of hardware technologies can be found at tabletop surfaces with technologies based on DI as the most versatile choice in terms of possible interaction techniques.

Multimodal
technologies.

Tangible technologies.

All the technologies for multimodal interaction discussed in ► Chapter 4 are also suitable to be utilized in MSEs. In MSEs, they even provide more potential for interaction techniques by combining multiple interactive surfaces to realize distributed multimodal interaction. For example, depth sensors to recognize hand gestures or body postures may be used for interactions with a digital tabletop as well as for interactions between tabletops and wall displays or between several mobile surfaces. Tangible interaction usually requires large horizontally positioned surfaces for stable placement of the tangibles. Vertical or tilted positioned surfaces would require additional mechanisms (e.g., approaches using magnetic holds) to keep the tangibles grounded on the surface.

Overall, interaction techniques that are possible through the hitherto discussed enabling technologies are also available for MSEs and serve as base enabling technologies.

5.3.2 Infrastructure Technologies

Infrastructure technologies serve as the basic requirement to establish a distributed and connected workspace and to enable synchronized and distributed interactions. They have to provide mechanisms for the multitude of interactive surfaces within an MSE to communicate, coordinate and exchange data with each other. The availability, quality, and reliability of those mechanisms depend on the capabilities of infrastructure hardware together with software.

5.3.2.1 Infrastructure Hardware

Hardware
technologies.

On the part of the hardware technology, virtually every interactive surface fully supports wireless technology for connectivity, which is standardized (IEEE 802.11) and thus provides full compatibility independent of concrete implementations. Static large or immobile interactive surfaces, such as tabletops, often support wired network connections, which are usually faster than wireless net-

⁶ <http://www.microsoft.com/microsoft-surface-hub>; formerly known as Perceptive Pixel Displays <https://support.microsoft.com/gp/perceptive-pixel>

works. In contrast, mobile interactive surfaces, such as smartphones or tablets, have to be wirelessly connected to the workspace, as they generally do not have a wired network connector. Design decisions on this part (e.g., integrating faster components) might positively influence the performance of interaction techniques. However, those decisions mostly do not affect the general feasibility of interaction techniques.

Design implications.

5.3.2.2 Infrastructure Software

The key infrastructure component to span a connected and synchronized workspace lies on the part of the software technology that locates, identifies, interconnects, and mediates all participating interactive surfaces. Design decisions on this part entail many implications for interaction design in terms of supported device platforms, possible interaction techniques, or performance and efficiency of interaction techniques [Nacenta, 2009, p.11].

Software technologies.

Design implications.

EARLY INFRASTRUCTURE SOLUTIONS. The research community early registered the importance of such infrastructure components as well as the lack of suitable software infrastructure support for interactive environments and addressed this issue with framework concepts and architectures. For example, Tandler [2000] analyzed the requirements of such software infrastructures and presented their software component "BEACH" (implemented in VisualWorks SmallTalk), which was designed to meet the identified requirements. In this vein, Ponnekanti et al. [2003] also identified software requirements and presented "iROS" (re-implemented for every platform), which they placed into the software class of middlewares. A different approach was addressed by Román et al. [2002] who proposed "Gaia" (implementation based on OMG CORBA⁷) as a so-called "meta-operating system". Gaia was designed as a middleware infrastructure that provided additional network services.

Infrastructure solutions have always been an interest of research.

However, those early work rather addressed ubiquitous computing environments, which are different from our defined MSEs. In particular, they do not address nowadays predominant touch-based interactive surfaces and heterogeneous device ecologies. For example, SmallTalk applications would hardly run on Google Android or Apple iOS devices, as there is no support for SmallTalk on such devices. The same applies to OMG CORBA-based infrastructures in addition to CORBA being a huge run-time and software overhead for small mobile devices. Software infrastructures that are designed for (always-on) desktop or server systems are often inappropriate for mobile devices, which may be arbitrarily turned on and off by users.

Early infrastructure components are not designed for MSEs.

⁷ <http://www.corba.org>

Recent software
infrastructures for
MSEs lacks support
for heterogeneity.

RECENT INFRASTRUCTURE SOLUTIONS. There are only a few works that explicitly target MSEs, for example, "Shared Substance" (implemented in Python⁸) by Gjerlufsen et al. [2011], "glueTK" (implementation based on Clutter toolkit⁹) by van de Camp and Stiefelhagen [2013], or "ZOIL" (implemented in C#, WPF) by Jetter [2013]. While those frameworks address MSEs, they miss sufficient support for nowadays heterogeneous device ecologies. For example, applications based on Python, WPF, or Clutter toolkit are restricted to the targeted platforms, which are quite different from nowadays platforms, such as Google Android or Apple iOS devices. Overall, heterogeneity of device platforms in MSEs is one of the big challenges of infrastructure frameworks, which has not been adequately addressed.

ISOLATED INFRASTRUCTURE SOLUTIONS. Due to missing suitable software infrastructure approaches and solutions, many researchers employed isolated solutions and developed customized applications for research projects and studies. Such customized applications are barely reusable for further studies and are coined to particular devices and according configurations.

⁸ <https://www.python.org>

⁹ <https://clutter-project.org>

5.4 Environs - Interactive Portal - Multi-Surface Environment Framework

Contribution Statement: In this section, we contribute to the trending research domain "multi-surface interaction" and in particular the challenging application of interactive real-time portals. We first analyze the requirements and challenges of appropriate infrastructures and present our reference framework called "Environs" that fully meets the outcome of the analysis. The reference framework Environs is publicly available^a under the open-source "Eclipse Public License" in order to foster MSE research as well as replication of research works. An earlier version of this section's content has been published as a peer-reviewed full paper [Dang and André, 2014] at the EICS (Engineering Interactive Computing Systems) 2014 conference.

<https://doi.org/10.1145/2607023.2607038>

We further devised a home automation concept using the Environs framework and presented the concept at the 9th Conference on Interactive Tabletops and Surfaces ITS 2014 [Jin et al., 2014].

<https://doi.org/10.1145/2669485.2669553>

Furthermore, we developed a shared energy visualization application for multi-surface environments based on the Environs framework, which realized real-time semantic overlays through interactive portals. The application was presented as a workshop contribution at the 15th IFIP TC.13 conference on Human-Computer Interaction INTERACT 2015 [Dang et al., 2015].

^a <http://hcm-lab.de/environs>

The analysis and discussion so far showed that MSEs are expected to become more and more commonplace and that research works aiming at such environments increased constantly during recent years. In consideration of nowadays trend in computing and computing devices, which favor surface interaction instead of conventional mouse and keyboard input, distributed applications and cross-device interactions in MSEs will likely constitute a substantial and prevalent part of applications and interaction techniques.

Developments and trends of recent research related to MSEs.

Increasing interest of research in infrastructures and frameworks.

Advantages of MSE frameworks.

Furthermore, the increasing amount of research targeting appropriate programming infrastructures and frameworks (cf. ► Figure 5.6) clearly shows the importance of such frameworks within the research community. In particular, in software engineering and development practice, frameworks or toolkits are regularly used for recurrent and/or general tasks. Hence, it's conceivable that this will also be the case for the development of MSE applications. For example, a multi-platform framework that automatically handles network and connectivity, or device management and communication would greatly help unburden developers from implementing the required logic for each supported platform and every new application, which is also prone to errors. Considering the high complexity induced by nowadays heterogeneous device ecologies, MSE framework concepts and architectures or concrete examples of such an MSE framework would be quite beneficial for application development, research studies, and reproduction of research results.

MSE frameworks and interaction channels/views.

In addition to providing basic features, MSE frameworks should also address and enable the different interaction channels as explored in ► Subsection 5.2.2.3. The most challenging channel for MSE applications is realized through wireless interactive video portals between interactive surfaces (cf. ► Figure 5.7), which is also a unique feature of the contribution in this section. Interactive por-

Figure 5.7:

Interactive real-time portals established from a large tabletop surface to multiple small mobile devices that are placed on the tabletop.



tals replicate part of one workspace to another workspace (usually) on a mobile surface, for instance, to enable a world-in-miniature view [Cheng et al., 2012], to create a virtual loupe [Voida et al., 2009], to enable collaborative visual exploration [McGrath et al., 2012], or, more generally, as an extension in collaborative interactive spaces [Ajaj et al., 2009, Dippon et al., 2012, Jetter et al., 2013]. The given recent research efforts show that there is a growing interest

of the HCI community in interactive video portals for interactive spaces or appropriate interaction techniques. To address the issues and shortcomings that we have outlined so far, we present a software framework called "Environs" aimed at alleviating the development of MSE applications. The framework supports nowadays heterogeneous device environments and particularly addresses low-latency and high-resolution video portals for interactive portal applications. It consists of self-contained platform-specific libraries that manage available application counterparts within the MSE, dynamically couple each other, establish interactive video portals, and enable user interaction and applications spanned across multiple displays.

MSE framework
Environs.

This section gives an overview of the Environs framework and answers questions that we encountered in designing and implementing the framework, for example, how to design the architecture/infrastructure, what components are required, how to couple application counterparts and establish communication, or how to distribute responsibilities over components. We also present two example applications that employ our framework to enable research for interactive video portal applications within MSEs. Even though parts of Environs provide service concepts typical of middlewares, we use the term framework for Environs because of the more general meaning of the term framework.

Structure and content
of this section.

5.4.1 Related Work

The research literature related to MSEs has been discussed in the previous ► Section 5.2, giving (1) an overview of the broad range of research topics and (2) means to categorize and sort them (Design Space in ► Table 5.1). The majority of MSE research works provide studies for interaction techniques (e.g., Bezerianos and Balakrishnan [2005], Jetter et al. [2013], Nacenta [2009], Seyed et al. [2013], Wilson and Benko [2010]), interaction metaphors and gestures (e.g., Seyed et al. [2012], Spindler et al. [2009], Wallace et al. [2006]), or concrete applications (e.g., Cheng et al. [2012], McGrath et al. [2012], Volda et al. [2009]). Research contributions related to MSE frameworks have already been discussed in ► Subsection 5.3.2.2. In this section, we concentrate on frameworks related to interactive video portals used in MSE applications. We start with early conceptual ideas and continue with research works up to recent approaches to realize interactive real-time video portals. Following up, we discuss works that investigate the most challenging aspect of such MSE applications, that is, research contributions regarding latency of visualization and interaction, which are considered individually as well as in concert with each other. Finally, we give an overview of selected frameworks and toolkits that (1) serve to enable applications and interaction techniques for MSEs and (2) discuss their applicability for interactive portals.

MSE related work in
► Section 5.2.

MSE framework
related work in
► Subsection 5.3.2.2.

We concentrate on
MSE frameworks and
applications related to
interactive portals.

5.4.1.1 Interactive Portals

A large body of research for interactive portals studied user interface and interaction metaphors, for example, a tool-glass [Bier et al., 1993], a peephole [Grubert et al., 2012, Henze and Boll, 2010, Yee, 2003], or a magic lens [Bier et al., 1993, Henze and Boll, 2010, Rohs et al., 2009]. They applied mobile augmented reality techniques to enable a portal, where the device's back-facing camera was used to capture the portal source in order to augment the captured portal and to ultimately display the result on the device display [Grubert et al., 2012, Henze and Boll, 2010, Rohs et al., 2009]. While this approach works well for the underlying metaphor, it also poses limitations to the applications. For example, navigation or zoom interaction is intrinsically tied to the mobile device's physical position and orientation due to the fixed mounting of the camera. Moving the mobile device immediately changes the perspective of the portal and its content, which makes it impossible to pass the real-time portal with the same content to collaborators after walking around to another location. Users instead have to take a picture of the content to achieve this goal. However, a picture is not a real-time portal anymore but static content, which falls short in always reflecting the actual state of the portal source.

Most implementations used mobile augmented reality for portal creation.

Limitations of mobile augmented reality.

Alternative portal approaches include having the whole portal content loaded in advance onto the mobile device as proposed by Yee [2003] or restricting the portal content to geometrical drawings as demonstrated by Holmquist et al. [2003]. Only a few realized an interactive portal for mobile devices that overcome the described limitations, for example, [Cheng et al., 2012, Jetter et al., 2013, Tsao et al., 2011, Volda et al., 2009]. However, they still suffer from restricted applicability for mobile devices. For example, Tsao et al. [2011] or Baudisch et al. [2001] facilitated interactive portals based on VNC¹⁰, which was originally designed to transmit screen captures on an event triggered request mechanism. Thus, VNC is not particularly adequate for real-time streaming of an interactive portal in video quality. The drawbacks in terms of latency, performance, resource usage, or bandwidth requirements become obvious and visible if the portal source includes a video clip or movie. Even though recent developments of VNC improved the approach in terms of quality and performance, its design and concept targets desktop or server systems where computing power and system resources are superior in comparison to mobile platforms. On mobile platforms, draining the battery as little as possible and gaining as much usage time as possible with the available resources are quite important principles in order to offer the best user experience and make an approach successful.

Interactive portals through preloaded and predefined content.

Interactive portals through VNC.

Limitations of VNC.

¹⁰<http://www.realvnc.com>

5.4.1.2 Latency

Besides visual quality and applicability of portal implementations, in particular for MSEs in the wild, latency is an equally important quality. In this work, we consider latency as the duration for changes of visual content on the portal source to become visible on the portal destination. While prior works neglected latency issues, the impact of latency on user interaction and user experience increasingly became the focus of attention of recent research efforts in other domains of HCI, for example, by Anderson et al. [2011], Bérard and Blanch [2013], Jota et al. [2013], He et al. [2000], or Ng et al. [2012]. In addition to bad user experience, they further emphasized the negative effects of high latency on task performance and error rates. Let's start with the commonly targeted latency of the "pre-surface" epoch, that is, the time before direct-touch interaction with surfaces (e.g., smartphones, tablets, or touch notebooks) became as important as today. Since the early work in 1968 by Miller [1968], the "100ms rule of thumb" has been widely asserted for an upper recommendation for GUI feedback to seem instantaneous, whereas the evolution of technology educed increasing performance gains of mobile processors and new forms of devices, applications, and corresponding interaction techniques. Consequently, researchers focused again on system latency. For example, Jota et al. [2013] studied the effect of latency in direct-touch pointing tasks and showed how task performance significantly decrease and error rates increase as latency increase. Thus, reinforcing an earlier Fitt's law study of MacKenzie and Ware [1993] who identified "*latency as a major bottleneck for usability*". Ng et al. [2012] proposed to explicitly consider latency in user interface design to cope with system latency and Anderson et al. [2011] identified an effect of latency on user experience where users perceive lower latency as more responsive. What we can conclude from recent findings is that portal implementations that suffer from high latency in visualization and interaction not only has a negative effect on task performance but also becomes annoying for users, which in turn declines user experience.

Latency is as important as visual quality and applicability.

High latency negatively affects user experience, task performance, and error rate.

The 100ms rule of thumb.

Refocus on latency in research.

5.4.2 Example Applications

Before describing concepts and technical details of Environs, this section aims at giving the reader an impression of what interactive real-time portals in MSEs are and what the Environs framework's functionality enable by depicting example applications and scenarios. Similar to the structure of the introduction to this chapter, these examples shall ease access to the content of the remaining sections. We have realized the following two example applications using the Environs framework to prove the usefulness of the framework's capabilities and its advantages in terms of easy integration as well as reduced development cost.

Two example applications to show what the Environs framework features.

Those applications also served to conduct research for appropriate interaction techniques in MSEs through interactive portals.

5.4.2.1 MediaBrowser

What is the aim of the MediaBrowser?

The MediaBrowser is a distributed application consisting of an application for tabletop surfaces and mobile devices. The applications are designed for collaborative reviewing or examining of media data on large tabletop displays. They aim at studying interaction techniques that best support collaborative tasks within such an interactive MSE scenario.

Figure 5.8:
Example application: Media Browser running on a large tabletop surface and multiple smaller mobile surfaces.



Usage scenario.

Users who run the MediaBrowser on their mobile device are first presented a list of available MediaBrowser devices and tabletops that were detected by the framework. The framework updates this list automatically, allowing users to participate in an ad-hoc fashion. Upon being presented with the list of application counterparts, users may transmit different kinds of media data, such as images or text-documents, with each other through the MediaBrowser. Media data transmitted to the tabletop are immediately shown on the tabletop display where all media objects can be manipulated through multi-touch input.

In ► Figure 5.8, the MediaBrowser shows multiple images that can be moved or scaled through multi-touch gestures. Bystanders who want to take part in the collaborative task just place their mobile device on the tabletop surface whereupon a video portal between the devices is automatically created. As depicted in ► Figure 5.7 and ► Figure 5.8, the mobile devices resemble transparent windows, which reveal the tabletop surface area occluded by the device. In order to detect mobile devices on the tabletop surface, every mobile device has a Microsoft Surface supported visual byte tag¹¹ attached at the backside (cf. ► Figure 5.9). The portal stays connected and updated even if a device is lift off from



Figure 5.9: A Microsoft Surface byte tag attached to the backside of a mobile device.

the tabletop surface, allowing users to virtually pick up a piece of the tabletop surface by means of their personal device. Users can further input multi-touch gestures on their mobile device, which are directly applied to the media on the tabletop surface. By this way, multiple users collaboratively interact with the tabletop surface in parallel without having the presentation of the large tabletop surface to suffer from space conflicts or occlusion issues due to too many arms and hands of collaborators. In addition to that, users are able to interact on the tabletop surface together with collaborators. Development of the application greatly took advantage of the Environs framework's functionality, allowing developers to focus mainly on user interface and presentation-related logic.

5.4.2.2 Public Display Toucher

The second example application "Public Display Toucher", as shown in ► Figure 5.10, consists of an application for public displays and mobile tablets. This application demonstrates how users may operate large public displays by means of a tablet's input capabilities. Users connect to a public display through an according tablet application, which enables them to transfer media data to the public display's desktop or operate through an interactive portal. Upon creation of an interactive portal, the portal's location and size can be adjusted through performing three finger multi-touch gestures. The public display can be controlled by means of single touches on the tablet, which are translated to mouse clicks on

Ad-hoc participation with mobile surfaces through intuitive gestures.

Interactive portals make devices translucent.

Multi-touch input on mobile devices conducted to the tabletop surface.

Collaborative usage.

What is the aim of the Public Display Toucher?

Usage scenario.

¹¹<http://www.microsoft.com/download/en/details.aspx?id=11029>



Figure 5.10: Example application: Public Display Toucher. Interaction with a large wall-sized display through personal tablet devices.

Touch input translated to mouse and keyboard input.

the public display's user interface. Furthermore, key input on the tablet's virtual keyboard is put through to the public display and translated into regular key events. By this means, the tablet takes over the public display's mouse and keyboard, allowing users to operate the public display's desktop, for instance, to start applications, perform mouse clicks, or enter text. Based on this basic functionality, multi-touch enabled applications may be started on the public display, which may be further controlled with the tablet. Just as with the MediaBrowser, development of the application mainly focused on user interface related logic. In addition, the application for the public display included logic for translating key messages from tablet devices into Microsoft Windows key events.

5.4.3 Requirements and Challenges for MSE frameworks

When engineering a framework for the MSEs in the focus of this chapter, questions arise such as, for example:

Questions regarding architecture, design, and implementation.

- How to design a framework to support different platforms without implementing, managing, and developing the whole framework for each platform separately?

- How to structure and distribute responsibilities for a reasonable architecture?
- How to manage the participating devices and applications in case of
 - multiple different MSE applications running in the same physical MSE?
 - or in different physical but logically connected MSEs?

We address such questions regarding architecture, design, and implementation by first identifying essential requirements on the MSE framework in question and then presenting our approach. The following identified requirements are also considered as challenges to be tackled within the engineering process:

- 1. Support for heterogeneity of platforms.** From a technical point of view, a big challenge for a framework is to support heterogeneous device platforms. For each platform, the framework's functionality has to be implemented based on platform-specific development requirements. For example, each platform requires developers to use a certain programming language, such as Java for Google Android, C# for Microsoft Surface / Windows Phone, Objective-C / Swift for Apple iOS, or C++/HTML5 for Tizen / Firefox / Sailfish OS. Moreover, each platform provides access to system and platform functionality through different APIs, paradigms, packages, and methods.

Different platforms require different implementations.

In the "pre-surface" epoch, a good option to fulfill this requirement for desktop computing systems was to base a framework on the Java infrastructure in order to enable cross-platform solutions. However, this is not an option for today's device ecologies of mobile surfaces where it's not even possible to have Java on some of the platforms.

Java misses multi-platform support for nowadays devices.

Supporting different device platforms is not only reasonable for commercial development but also for scientific research in case of distributing the framework or framework-based applications to fellow researchers who may not necessarily use the same device platforms.

- 2. Efficiency and latency.** A framework must provide efficiency and low latency for network transfers and for framework logic. In particular, video portals require fast packet transfers and fast processing of the video stream in order to enable low latency. The lower the latency, the more responsive an application appears, which directly affects user experience, applicability, acceptance, and success as previously discussed.

Network and framework latency. Responsiveness.

- 3. Flexible network connectivity and device management.** Finding and managing available devices must support MSE scenarios in which devices

Ad-hoc network behavior.

take part in an ad-hoc manner and may vanish suddenly, which is the regular behavior of mobile surfaces. Devices have to identify themselves to each other and approve or deny connection requests as well as handle connections from multiple devices in parallel.

Complex physical
network layouts.
Multiple network links.

Furthermore, MSEs in the wild often have complex network and device behaviors. By "in the wild", we mean real-life environments that are not laboriously prepared to meet clearly defined conditions, that is, the opposite of a "sterile" lab situation for an experiment. There may be environments with mobile data only, with wireless network but no internet access, or multiple logically separated / connected subnets with or without internet access. Usually, today's computing devices are equipped with multiple network interfaces (WiFi, Mobile data) and large surfaces even have multiple interfaces of the same type (e.g., multiple LAN interfaces), which further increases the complexity of managing device connectivity within MSEs.

Novel interaction
techniques using
mobile sensors.

- 4. Sensor and user input support.** Nowadays devices are richly equipped with sensors, such as touch sensor, accelerometer, compass, or gyroscope. They enable novel interaction experiences if they are integrated into the design of interaction techniques as demonstrated by the many works in ► Table 5.1. Hence, support for sensor input is a strong requirement for frameworks that targets MSE applications. A framework has to provide the support and infrastructure to retrieve, transport, process, or consume sensor data. In particular sensor data that contribute to interaction related logic must be handled with higher priority than application logic when transported over network channels or processed within the framework logic.

5.4.4 Design of Environs

Design goals for
Environs.

Environs is a software framework designed for the development of distributed applications for MSEs with support for interactive low-latency video portals. The whole design and architecture were developed with a strong focus on the aforementioned challenges (and requirements), that is, support for nowadays heterogeneous device platforms, ad-hoc networking connectivity, sensor data support, while having as low latency as possible.

Supported platforms.

Developers of such applications are given a set of self-contained platform-specific libraries of which developers only have to include the libraries that target their desired device platforms. In the current development state, Environs supports the platforms Apple OSX and iOS, Microsoft Windows, Google Android, Linux, and Raspberry PI.

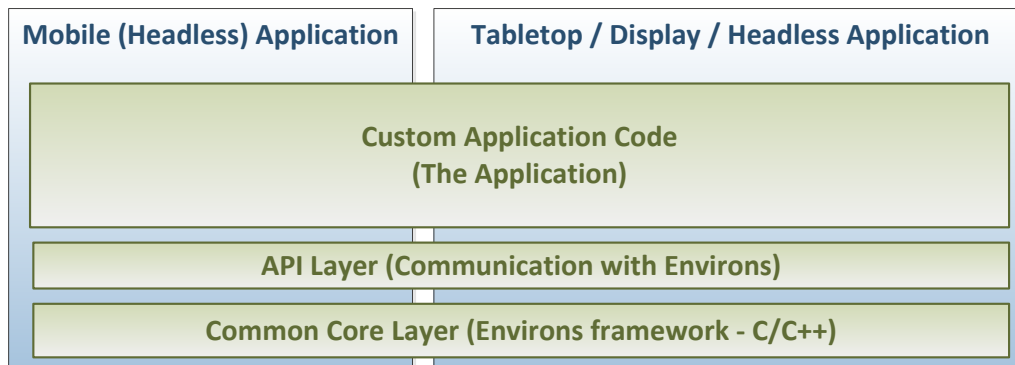


Figure 5.11: Layered design of applications that make use of the Environs framework.

ACCESS TO FRAMEWORK FUNCTIONALITY. Environs' functionality is designed to be implementation agnostic to custom application logic, which means that, for example, applications don't have to care about how to detect other devices, reach them, exchange messages, or transfer files to them. Applications only need to invoke calls of the framework's API that can succeed or fail. The framework's functionality so far include device and environment management, management of connections to other devices, transfer of files or binary data, communication between application instances by text or binary messages, real-time streaming of touch contacts to other devices and conducting those touch contacts on destination devices, real-time streaming of sensor data to other devices, gesture recognition using touch and sensor data, and real-time streaming of customizable high resolution video portals. Every framework library provides three consistent APIs across the supported platforms for accessing the functionality. Developers have the choice between a pure C-API, an event-driven static (raw) API, and an object-oriented API, which will be presented in ► Subsection 5.4.7. The design of the Environs framework libraries and applications that are based on the framework distinguishes between two types of applications and three logically separated vertical layers as depicted in ► Figure 5.11.

How applications make use of Environs.

Features of Environs.

Unified API across all platforms.

5.4.4.1 Application Types

► Figure 5.11 indicates through a horizontal separation that applications can either target *mobile surfaces* (e.g., smartwatches, smartphones, tablets, touch notebooks) or *large surface devices* (e.g., tabletops, floor displays, or wall displays). The application type defines a base behavior of the framework in terms of dealing with available resources or the handling of sensor data. For example, while memory allocations for applications on desktop operating systems are only limited by the available RAM (it may grow above 4 GB), it's quite different for mobile operating systems. On mobile systems, the allowed size of memory allocations by one application (and process) is considerably less and typically range from 16 MB up to around 256 MB depending on the platform and configuration

Application types: mobile surfaces, large surface devices.

Application types predefine the base behavior of the framework.

Application type as a first step to approach heterogeneity.

of a device (even if the device is equipped with much more RAM). Designing a framework without considering such differences might lead to an unusable framework as the framework logic would claim too much memory on mobile systems, which in turn results in insufficient available memory for the actual application.

Application of headless mode.

HEADLESS MODE. Both application types can be further distinguished as applications with user interface or headless mode applications. Environs drops support for user interfaces in the headless mode, which is useful, for example, for applications that employ their own user interface technology, such as surface games based on the game development platform Unity¹². In such cases, providing support for user interface related logic is rather disadvantageous, as it would conflict with the technology used by the application development platform or claim precious resources for redundant functionality. Another useful application domain for headless mode is realizing services in general. For example, services that provide location-based information about a physical environment or services running on small devices (e.g., Raspberry PI) that take part in IoT applications. Such applications may or may not have a user interface, thus would require either user interface support or no support for user interfaces.

Avoid conflicts or redundancy in user interface logic.

Build MSE / IoT services without user interfaces.

5.4.4.2 Application and Framework Layers

► Figure 5.12 shows the vertically distributed application and framework layers of ► Figure 5.11 enriched with examples and more details on the different layers, that is custom application layer, platform-specific API layer, and common core framework layer.

Application development by developers is represented in the custom application layer.

APPLICATION LAYER. The Application layer represents the actual application that users interact with, that is usually user interface related code, application logic, and data management and storage logic. The complexity and size of this layer greatly depend on the functionality and presentation provided by applications. On mobile device platforms, application development is usually constrained to certain development tools and programming languages, for example Objective-C / Swift for Apple iOS or Java for Google Android. In most of the cases, it's best to employ the development tools and infrastructure provided by the platform manufacturers in order to achieve the best performance as well as user experience for an application. Therefore, Environs supports application development using the designated development tools, that is, Google Android using Java, Apple OSX and iOS using Objective-C, Microsoft Windows using C# and C++, and Linux / Raspberry PI/ Apple OSX using C++ as shown in ► Figure 5.12.

Application development supports platform tools provided by manufacturers.

¹²<http://unity3d.com>

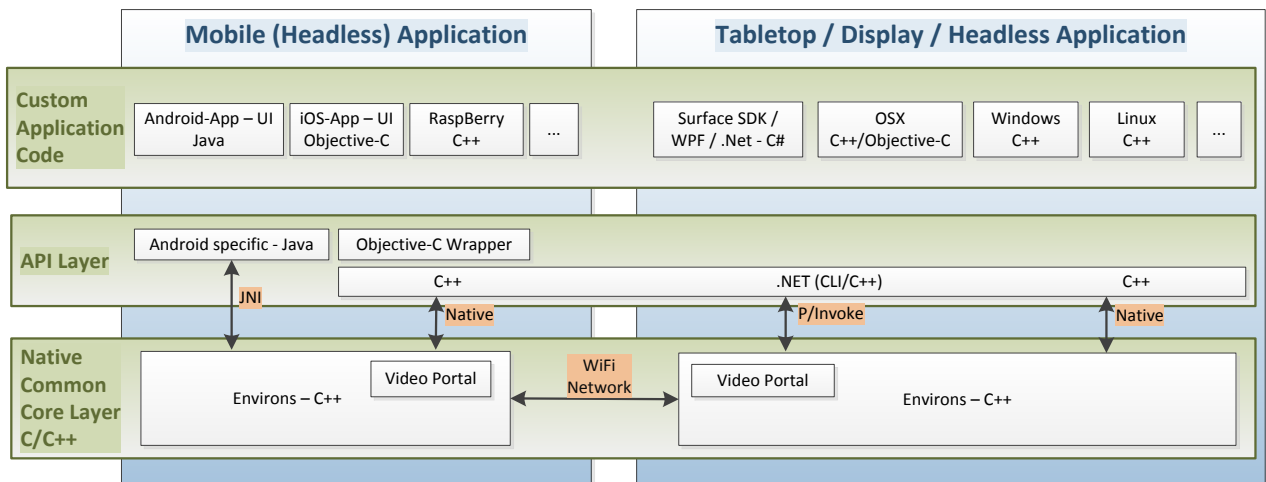


Figure 5.12: Layers of applications using the Environs framework enriched with details and examples.

PLATFORM SPECIFIC API LAYER. The Platform-specific API layer is a thin layer, which acts as the glue (also called binding in programming language usage) between the native core layer and the application layer and provides access to the framework's functionality through an API that makes use of the platform-specific programming language, paradigms, and resource management. By this means, the Environs framework enables seamless integration of the framework functionality into custom application layer logic. Detailed descriptions for the platform-specific API layer will be given in ► Subsection 5.4.6.8.

Platform API layer as glue / binding between application logic and framework logic.

NATIVE COMMON CORE LAYER. The native common core layer is the largest part (90%-95% of lines of code) of the Environs framework and implements the whole functionality. All platform libraries share the same code-base for the common core layer, which is written in portable plain C and C++11. For each supported platform, this core layer is compiled as a native library and loaded by the platform-specific layer during run-time.

Core layer implementing the framework functionality.

The base behavior of the core layer is determined by the target device type, that is, whether the code-base is built for a mobile surface or a large static surface. Later at run-time, the core layer queries details of the particular platform on which the code is actually executed in order to dynamically fine tune the behavior, for example, in terms of limits for device connections or object instances, resource allocations, usage and size of caches, buffer sizes, or growth sizes of buffers. This is particularly important for mobile surfaces, as available resources on such devices vary quite a lot. An average framework memory footprint of 1 MB is more than okay if memory allocations are allowed up to 256 MB. However, if the overall allocations are only allowed up to 16 MB, then 1

Base behavior defined by application type. Dynamic adaptation of behaviors at runtime.

MB would claim a considerable amount of the available heap memory space. For such devices, the average memory footprint must be minimized as best as possible. ► Subsection 5.4.5 will give an extensive overview of the architecture, that is, the components and how they relate to each other, as well as important design decisions.

5.4.4.3 Code Distribution

Code distribution as an approach to tackle heterogeneity of platforms (challenge 1).

Advantages of code distribution.

The code distribution of Environs presented so far addresses the *challenge 1 (heterogeneity of platforms)* and allows development of the majority of the framework to focus on the common code-base for all platforms, for instance adding new features or extending available features, modifying transport protocols, or debugging. The common code-base drastically minimizes development costs required for maintenance and development whilst supporting heterogeneous device platforms. Instead of implementing the logic for each platform separately using the programming language that is destined for the particular platform, developers merely have to put the effort on only one common code-base while benefit from manageable and reduced development time.

Many platform compilers.

Many platform tools.

ROBUSTNESS OF CODE. This approach further enables development of quite robust code because of (at least) two reasons. Firstly, the same code-base is compiled and therefore analyzed and verified through powerful tools of many platforms (e.g., Microsoft Visual Studio, Apple Xcode, clang, or GCC). It is comparable to viewing the implemented logic and code traces from many different (compiler) perspectives, which reveals more errors, mistakes or flaws that have been overlooked during development if taken only one target platform or compiler into account. Secondly, every platform development environment further offers tools for profiling and analyzing (e.g., Microsoft Visual Studio Profiler, Apple Xcode Profiler/Instruments, or Valgrind¹³) that greatly help stabilize and increase the robustness of the framework logic. We found that employing tools from different platforms together revealed much more defects than relying on only one of them. This is because each tool has its strengths and weaknesses and focus on different aspects of code quality. (For example, the Microsoft Application Verifier offers effective approaches to analyze misuses of network sockets, which the other tools hardly can match up with, while Apple Instruments offers quite advanced mechanisms for tracing of memory allocations.)

Addressing high performance and low latency (challenge 2).

PERFORMANCE AND OPTIMIZATIONS. In addition to low development effort and robust framework code, native code provides rich opportunities to realize high-performance implementations and low latency optimizations, for example, in terms of memory management, platform optimizations, or direct access to

¹³<http://valgrind.org>

hardware components, which addresses the *challenge 2 (efficiency and latency)*. Especially interactive video portals require processing and transporting of a large amount of video data within extremely short time frames. Such tasks greatly benefit from direct control over memory within the native layer, for example, to avoid unnecessary copying of video data multiple times or processing/transforming the data between the same/reused buffers instead of multiple different buffers. Reusing the same buffers helps exploit the sophisticated multi-layer cache hierarchies of modern processors, which are by multiple magnitudes faster than the access to the main memory.

Foster usage of modern processor caches.

5.4.4.4 Design Considerations for Mobile and Large Surface Computing Environments and Platforms

Design decisions throughout the framework from the architecture of components and layers in general up to concrete code layer related decisions were guided by a few simple but highly influential rules with an impact on how the framework and API works. Those rules are related to requirements on software for mobile and large surface computing platforms that emerge from user expectations on such software applications. Surface computing platforms and environments vary widely in hardware specification and most are quite different from traditional desktop workstations. Therefore, an MSE framework has to guarantee full operability of all functionality at least on small mobile surfaces, which influences design decisions, for example, in terms of resource usage, memory footprint, or usage of processing units.

Simple rules for design decisions with high impact.

Full functionality on small mobile devices as lowest denominator.

The most limiting resource of mobile surfaces is available battery capacity. Hence, design decisions have to address this limitation at all. The longer a user is able to use the surface device within the MSE, the better the user experience of an MSE application becomes. Using a device as long as possible until it needs to be recharged has become an important factor for software and hardware with the advent of mobile touch-devices. Hardware designers address increasing user expectations on this property by new device designs such as ultrabooks or netbooks. Furthermore, modern platform designs support sophisticated mechanisms, which put the whole system or parts of the system into deep sleep states to reduce battery consumption. Using such mechanisms or not using such mechanisms can make a difference between a few hours and several days of usage time. Therefore, design decisions must consider and foster the usage of such mechanisms.

Each design decision must aim at preserving battery capacity.

The architecture and components of Environs address this by allowing the platform to suspend as often and as long as possible. All design decisions within the framework incorporate this rule. For instance, instead of having one pro-

Examples of how design decisions address battery consumption.

Synchronization favors lock-free approaches.

Asynchronous API allowing application code to optimize resource usage or suspend whenever possible.

cessing thread handling multiple tasks one after another, Environs exploits parallelism as best as possible or uses multiple threads to perform tasks in order to keep the total "active" time as short as possible. One implication of highly parallel code execution is the requirement for synchronization mechanisms due to concurrent access to shared resources, for example mutual exclusive (mutex) or locked (critical sections) access to blocks of code. Such mechanisms, however, could potentially block parallel threads from each other – if not carefully implemented – which reduces the efficiency of parallel execution. Therefore, all design decisions throughout Environs favor so-called lock-free or wait-free approaches (Herlihy [1993], Barnes [1993]) over mutual exclusion or locks.

Another design rule is reflected in the APIs, which are designed to work asynchronously if tasks would take more time or would potentially block the caller. For example, network related operations – such as sending a message and waiting for acknowledgment – may block the caller for an unpredictable amount of time as there is no guarantee that the receiving entity in the network or over the internet is responding immediately or even accessible. In such cases, application code that makes calls to the APIs are not blocked in order to allow application code to immediately continue with application logic instead of maintaining a multitude of threads or blocking the device from suspending. Results of such an operation are reported back asynchronously through a notification subsystem of Environs. However, API calls can be instructed to work synchronously if application code requires such a code flow.

5.4.4.5 Concept of Application Environments

Different physical locations.

Logically separated MSEs.

The Environs framework enables different combinations of separation and visibility of MSEs, which are sketched in ► Figure 5.13. MSEs may be hosted in different physical locations as exemplarily denoted with "Physical Location" 1 and 2 in ► Figure 5.13. Within each physical location, an arbitrary amount of logically separated MSE may coexist at the same time. For example, a lab at a university may have an MSE for the meeting room as well as a logically separated MSE ("University") for public displays. Even though both MSEs are in the same physical building or local area, it makes sense to separate them logically as applications targeting the meeting room should not be disturbed by applications that interact with public displays outside the meeting room.

Logically connected MSE across multiple physical locations.

An MSE may also be spanned across multiple physical locations as sketched with the MSE called "University" in ► Figure 5.13. For example, an MSE application to interact with public displays that are distributed over the whole university campus would require access to displays located at many places. In such an MSE, it would also make sense to limit certain applications to certain physical

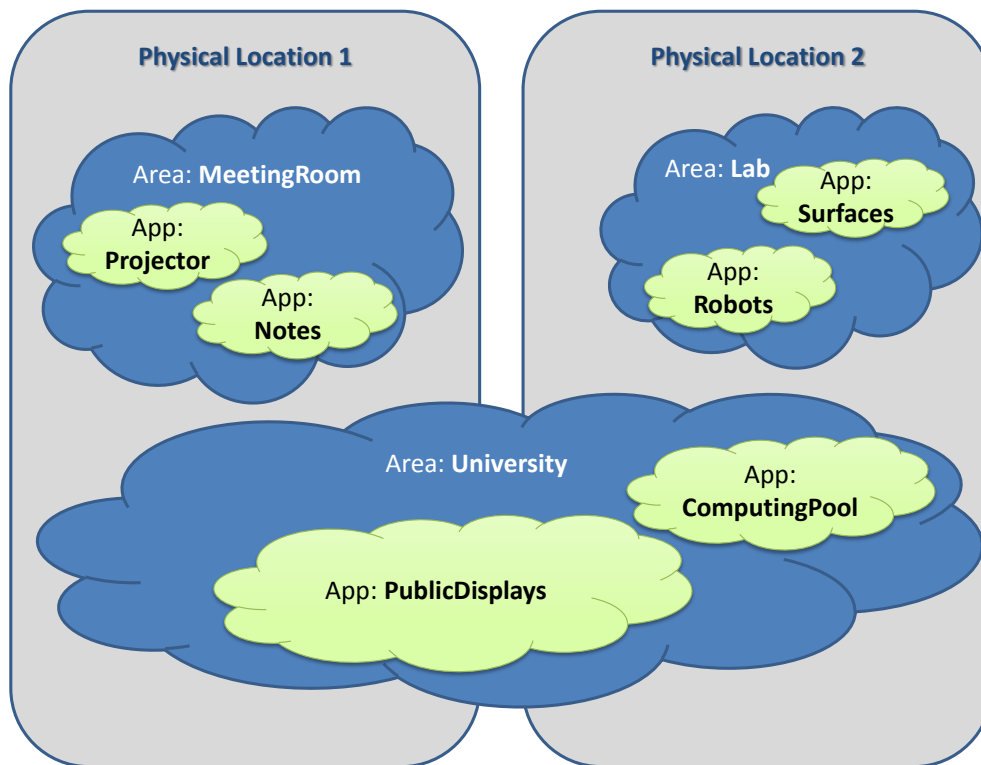


Figure 5.13: Multiple application environments in different physical locations distinguished by area / application names. One and the same application environment (e.g., PublicDisplays) may also be spanned across multiple different physical locations.

locations. For example, it is reasonable to restrict the application "Computing-Pool" in ► Figure 5.13 to a particular room of the MSE "University".

In order to enable all those scenarios and at the same time provide easy access to control separation and visibility, Environs makes use of *application environments*. An application environment is a logically connected workspace, which is spanned across the physical locations defined by an area. The Environs framework manages connectivity between multiple physical locations automatically without further consideration by custom application logic. Applications may choose to limit an application environment to a particular physical location, for example by providing network address filters.

Application
Environments.

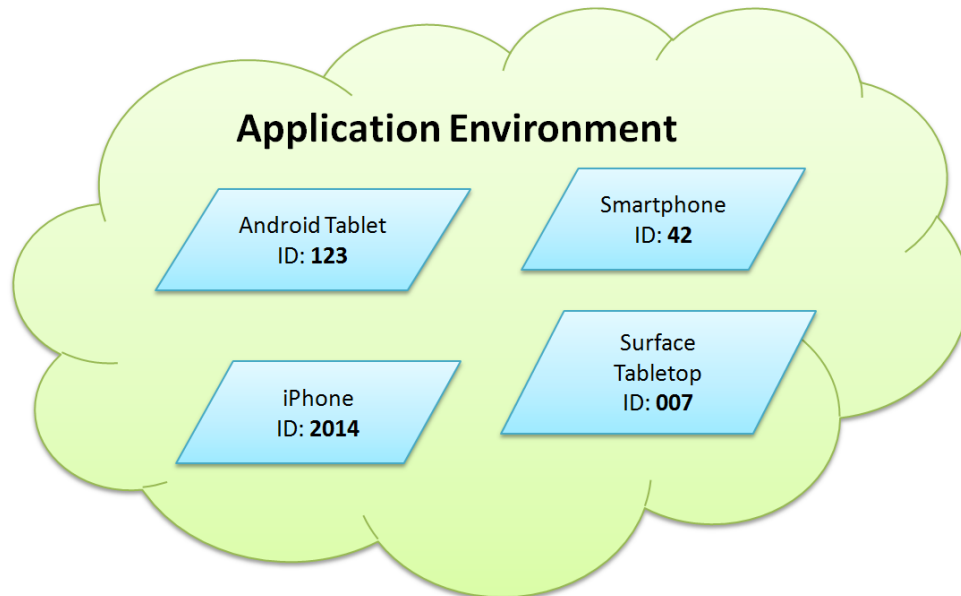
Each application environment is identified through an area name and an application name, which allows scenarios where, for example, multiple different (and logically separated) Environs-enabled applications run on the same tabletop display while the framework guarantees that each application sees and interacts only with other corresponding applications running within the same MSE. This logical separation helps the application logic of custom applications in communication with other application instances because different applications potentially communicate with disparate communication protocols. For example, a tablet application that exchanges videos or images with a tabletop display would not need to connect to a public display application in the same MSE that runs a blackboard application.

Application
environment = Area
name + Application
name.

Environs guarantees
logical separation of
different application
environments.

Figure 5.14:

Device IDs in application environments. Addressing of other application instances happens by means of area name, app name, and device ID.



Each application can request to only see and communicate with applications either with matching area and application name, or with matching area name. The additional area name enables hybrid applications that provide the functionality of multiple different applications together and therefore need to see application instances of all application names within the same area.

In order to identify devices within an application environment, each device is assigned an ID (numeric 32-bit value) as sketched in ► Figure 5.14. This ID must be unique only within the same application environment. In particular, the same device ID may be invalid for a different application environment if the device ID has already been claimed by an existing device. The concrete assignment of IDs or partitioning of ID ranges can be chosen depending on particular applications. For example, IDs lower than 1000 are assigned to tabletops and IDs greater than 1000 are assigned to mobile devices.

Device IDs must be unique in an application environment.

Automatic assignment of device IDs by Environs.

If no application specific partitioning or assignment of IDs is required, applications may leave the assignment of IDs to the framework and start with an ID of 0. The Environs framework then automatically detects which IDs have already been assigned within the MSE and which IDs are available for assignment. This automatic assignment behaves similar to requesting a network IP from a DHCP (Dynamic Host Configuration Protocol) server. In contrast to the DHCP protocol, Environs does not require a central server but relies on loosely coupled peer-to-peer communication.

5.4.5 Architecture of Environs

The architecture of Environs aims at scalability and adaptability to different platforms in order to support a wide range of device types from small mobile surfaces up to large interactive surfaces. To achieve this, Environs follows a layered architecture where each layer can scale up and adapt to the actual available resources depending on the actual platform at run-time. In prior versions of Environs (and related publication), the architecture of Environs followed a component-based approach where responsibilities were tied to dedicated components. Adaptation to different platforms involved substitution of the components with adapted platform components. However, in the course of development of Environs, it turned out that a layered approach is more flexible in terms of scalability and adaptability and suits the needs of Environs better. The layered approach allows distribution of responsibilities over different places within the code-base, thus, enable consideration of scalability and adaptability at many places while taking the code execution context into account. Some of the responsibilities are (still) bundled as components (e.g., notification broker or dynamic loader component) if the responsibilities can be cleanly encapsulated into a component.

What are the aims of the architecture of Environs?

Advantages of a layered architecture.

5.4.5.1 Environs Instances

► Subsection 5.4.4.5 reasoned by means of simple example scenarios why it might be necessary to have multiple application environments within the same physical environment or one logically connected application environment spanned across multiple physical environments. Multiple application environments that co-exist at the same physical location might be used by multiple different MSE application types whereof each type is assigned its own application environment. This can be implemented easily with an instance of the framework on each device that takes part in the particular application environment.

One Environs instance in an application.

Environs also supports more complex MSE scenarios in which it makes sense to run multiple instances of Environs at the same time within the same application instance to take part in multiple logically separated MSEs. Let's consider an example scenario in which several dedicated location nodes are installed in the same physical environment in order to collect location information about users and devices and provide those as a service within the MSE to applications. The location nodes shall further employ depth cameras (e.g., Microsoft Kinect) to sense the physical environment for users and their devices. Instead of operating as isolated tracking instances, the location nodes work together as a connected array of nodes, which share and synchronize their sensed information with each other. For example, they may perform sensor fusion approaches that include

Multiple instances within one application.

Example of using multiple Environs instances.

sensed data from all depth sensors or may perform handover detection from one region to another region of the physical space.

MSE applications on user devices that utilize such a location service usually do not need to take part in communications that happen between the location nodes. It would be even distracting for user applications to see the location service's details and each location node or related computing device. In such a scenario, it is reasonable to put the location nodes and all related communications into a logically separated application environment in order to keep them hidden from user applications. At the same time, however, at least one (or more) location service related node needs to be in the application environment in which user applications reside to offer the location service. This scenario can be realized by employing two instances of Environs on the location nodes to establish two logically separated application environments. One application environment serves for communications with other location nodes and the second one for providing the location service to user applications. The architecture of Environs enables running as many instances of Environs (and related application environments) as resources are available within the same application process. All instances work logically isolated from each other but are synchronized where necessary, for example when the application status changes and instances have to pause or get disposed. Since all instances run within the same application memory space, applications (such as location nodes) have access to multiple MSEs at the same time, for example, to implement the previously discussed example scenario.

Multiple instances of Environs to separate location node communication and user device communication.

Environs enables running multiple instances within the same application at the same time.

5.4.6 Framework Layers

► Figure 5.15 gives an overview of the layers and components that exist within the Environs framework architecture. There is one vertical layer (Core), six horizontal layers, and three components. All of them are part of one Environs instance, which in turn implies that each Environs instance consists of those layers if multiple instances of Environs exist.

5.4.6.1 Dynamic Loader.

Dynamically loading of framework modules at runtime to tackle challenge 1.

The dynamic loader exists only once within the framework and serves all instances of Environs. Dynamically loading of framework modules is part of the approach to tackle challenge 1 (cf. ► Subsection 5.4.3) with the aim to support heterogeneous platform environments. The actual platform and available resources of a device at run-time can vary widely, for instance, in terms of the actual operating system and version (e.g., Microsoft Windows 32-Bit, 64-Bit, XP, Vista, 7, 8, 10; Google Android 32-Bit, 64-Bit, APIs), available sensors, (hard-

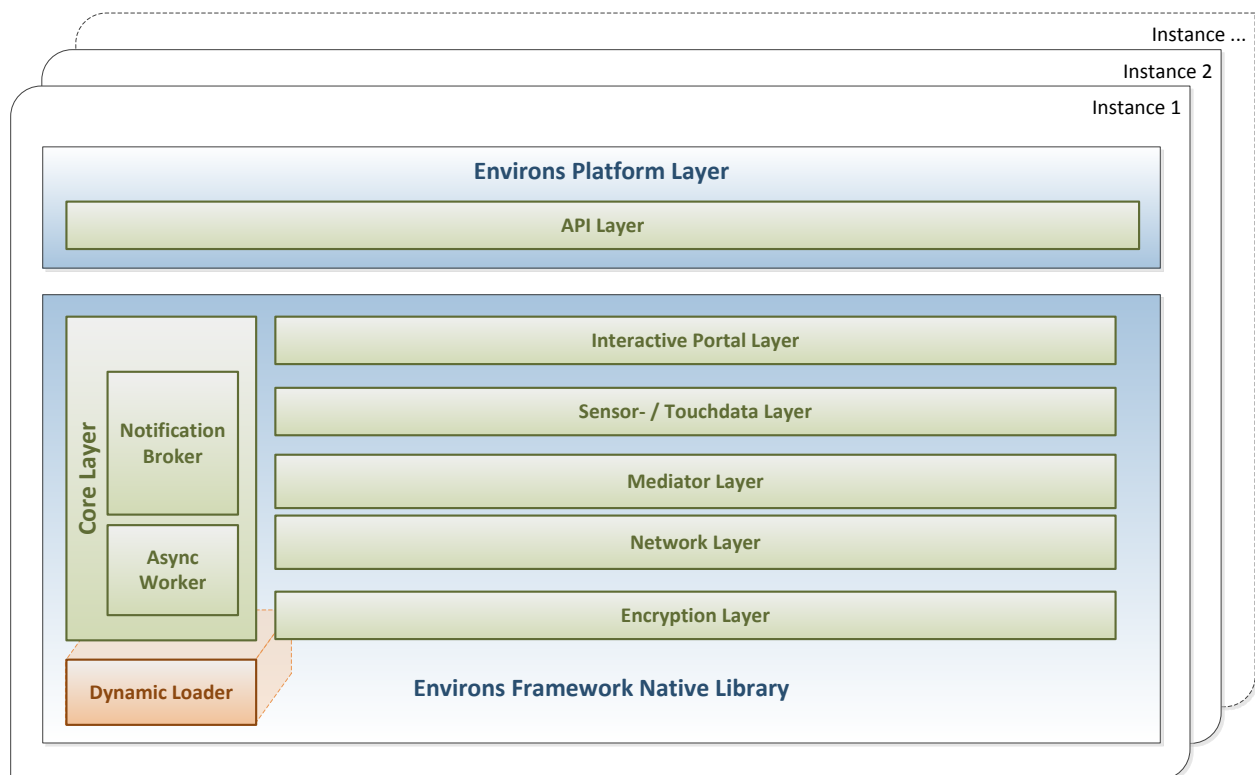


Figure 5.15: Architecture of Environs with its layers and components. Multiple instances with each having those layers can be running at the same time.

ware) support for encryption, hardware access to cameras or video encoder/decoder, or even the C-runtime libraries. At this point, the dynamic loader comes into play, which is utilized by all layers to determine the availability of resources at run-time and load modules that are most appropriate.

Furthermore, the dynamic loader helps solve issues due to fragmentation of operating systems (e.g., Google Android) and different versions of system libraries through dynamically querying of functions within system libraries instead of linking the framework against system libraries. If linked against system libraries, the framework would not be able to work if the correct version of a system library is not available. Dynamically querying of functions enables the framework to work even if a different version of a system library is installed. The dynamic loader enables the same framework binary to work with different versions of system libraries and to adapt to the actually available version. For example, the encryption layer makes use of the functionality of the OpenSSL¹⁴ library (if available) and adapts calls for initialization and release of resources depending on the available version of the library.

Dynamically loading framework modules to address fragmentation and different versions of system functionality.

¹⁴<https://www.openssl.org>

Dynamic loader as
bootstrap on Microsoft
Windows platforms.

On Microsoft Windows platforms, the dynamic loader is even the first entity of the framework that is executed in order to load the remaining framework libraries that are appropriate for the actual run-time environment. This is due to the wide span of native runtime environments of Microsoft Windows platforms, for instance combinations of C-runtime (aka platform toolset) and architecture (32- / 64-bit). Instead of constraining the framework's support to only recent platforms (e.g., only Windows 8.1 + 10 / 64-bit), the dynamic loader enables pre-building of Environs libraries for a wide range of platforms and dynamically loading the appropriate ones at run-time. All those loading steps are done automatically by the framework without explicit consideration by developers of user applications.

5.4.6.2 Core Layer

The core layer as
lifecycle maintainer for
other layers and
related objects.

The core layer
mediating
asynchronous calls
and callbacks.

The core layer is spanned vertically over all horizontal layers except for the API layer. The vertical spanning of the core layer denotes its responsibility for maintaining the lifecycle of the horizontal layers, that is, creating, starting, pausing, stopping, or disposing of the layers. That is, the core logic controls the horizontal layers to safely perform transitions between framework states. Furthermore, the core layer hosts the notification broker and async-worker component and manages their lifecycle the same ways as for the horizontal layers. Both components are key players for enabling fast and efficient asynchronous API access. Therefore, we show the interplay of both components with the API layer before we continue with the remaining layers.

Synchronous calls.

Asynchronous calls.

The arrows in ► Figure 5.16 sketch two types of access to the Environs framework through the API layer. The first one is access by synchronous calls denoted with S1 as the request and S2 as the response. In this mode, the control flow is passed on to the framework until the operation has finished upon which the control flow moves back to the calling entity together with the final results of the operation. This access mode is often preferred by developers due to its sequential control flow, which makes it easy to understand and which enables straightforward usage in user application code. Hence, this access mode is used whenever possible and reasonable, for example, if an operation can be completed without blocking the control flow (e.g., read/set an option, enable/disable a feature, or adjusting values). However, many API calls of the framework could potentially take longer and would block the caller for an unpredictable amount of time. The caller might be blocked for a relatively short duration (e.g., acquiring a mutex or wait until it's safe to enter a critical section) or might be blocked forever or until a timeout expires (e.g., network stuck during send of a message). For such API calls, the framework employs an asynchronous control flow by default as denoted with A1 to A6 in ► Figure 5.16. (Though, the caller may instruct

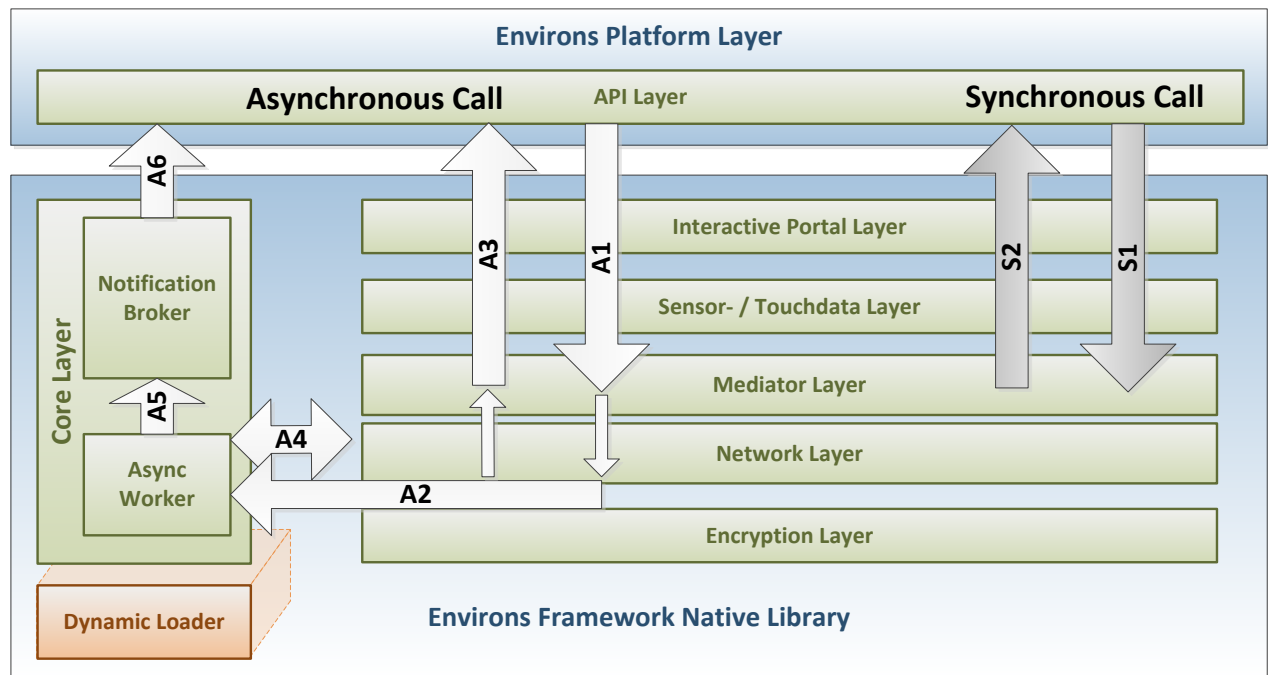


Figure 5.16: Steps of asynchronous (A*) and synchronous (S*) API calls.

the API layer to use a synchronous control flow.) The asynchronous call starts with the API request (A1) and passes the control flow to the framework logic, which in turn dispatches the request to the async-worker component (A2) and returns the control flow back to the caller (A3). At this point, the application logic can do other stuff or suspend the application. The result and intermediate states of the submitted request will be notified by the framework when it's done.

This kind of control flow is typically not sequential and for this reason not so straightforward to overview. The control flow rather stops at one place within the code and continues later at a different place within the code. An appropriate abstraction approach to keep the overview over the distributed control flow units is based on finite state machines comprised of a set of defined states and transitions between the states. The Environs framework internally makes use of state machines for every framework layer and almost every object. Some of the state machine representations are passed through to application logic by means of the API layer, for example Environs instances, device lists, devices, or interactive video portals. By submitting a request through the API layer, application logic instructs the framework to conduct a transition from the current state of a particular state machine to a target state. This is the point where one of the distributed control flow units stops. Once the state transition was successful, application logic can continue with the next transition, which represents the start of another

Core and layer objects managed through finite state machines.

Transitions between finite state machine states.

one of the distributed control flow units. For instance, by calling the method *Start ()* of an *Environs* instance, the instance performs a transition of the alive-state from stopped to started. Each transition has prerequisites that must be met by the current state machine's state. An *Environs* instance, for example, can only be stopped if the instance has been started before and vice versa. Other examples are states of devices that are available within the MSE. Those devices' state machines can be brought to a connected state or disconnected state. The framework entity that (asynchronously) conducts and coordinates those transitions is the *async-worker*.

Async-worker
thread-pool.

ASYNC-WORKER. The *async-worker* manages a FIFO¹⁵-queue with work-items that are dispatched by API calls or by framework layers. It is considered as the "work-horse" within the framework and serves a simple usage mantra:

"Whatever (task or operation) cannot be completed without blocking and I'm not allowed to wait for the result, let the work-horse do the job and get back to me when it's done".

Scalability of
async-worker.

Async-worker
thread-pool.

Therefore, the *async-worker* is designed to be highly scalable (to keep up with potentially high workload) and versatile in terms of work-items that can be dispatched. Work-items are classified according to their type in order to determine whether network access would be required or other types of blocking (such as accessing critical sections) would be involved. Depending on this classification, work-items are treated differently by the *async-worker*. In order to drain the *async-worker* queue and complete the work-items as efficient and as fast as possible, the *async-worker* manages a thread-pool with up to 12 - 32 (or more) threads that process the work-items in parallel depending on the actual device type, the available resources (e.g., CPU-cores or amount of memory) and the amount and type of work-items. In particular, work-items that entail network related operations are processed with dedicated threads as network access may block unpredictably. If not processed with dedicated threads, such operations would suddenly stop the whole flow of the *async-worker* preventing it from processing all other work-items, which might be finished within the same time that the network access would take. For example, connecting to a network device might take only 3 seconds while the CPU is idling and waiting for responses of the network peer, but within 3 seconds hundreds or thousands of non-network related work-items could be successfully completed. In order to avoid such worst-case scenarios, the *async-worker* always keeps one worker thread available for non-network related operations. Most of the time, the *async-worker* runs merely one thread to drain the queue. Additional worker threads are only spawned on demand (e.g., for network related work-items or if the size of the queue increase

Async-worker queue
drain strategy.

¹⁵First-In First-Out

rapidly, which would need support by more "work-horses") and destroyed as soon as possible. Depending on the actual work, the worker threads utilize one or more of the framework layers to complete a work-item, which is sketched with the arrow A4 in ► Figure 5.16.

NOTIFICATION-BROKER. As soon as the async-worker has completed a work-item, the results of the work-item are handed over to the notification-broker (A5 in ► Figure 5.16) in order to be presented to the application logic (A6 in ► Figure 5.16). Similar to the async-worker, the notification-broker manages a queue with notifications to work on and is designed to be versatile in terms of types of notifications. Reporting the result of an API request back to application logic happens most often due to successful or failed transition of the underlying state machine and represents merely one of the notification types.

Notification-broker.
The fast lane to
application logic.

Every layer of the framework utilizes the notification-broker to notify application logic about, for instance, framework states (e.g., an Environs instance has started or stopped), network states (e.g., network became lost or available), intermediate states (e.g., the progress of an ongoing connect to a device or the progress of a large file transfer to a device), or new events or data (e.g., a message from a device has arrived or sensor data has arrived). State machine transitions can happen not only due to API requests, but also due to external or internal framework events. A connection with another device or messages from other devices may be initiated by the other devices and network availability may change at any time without being caused by application logic.

Notification-broker
utilized by all layers.

The notification-broker guarantees that each notification is reported to application logic in the order of their chronological occurrence. This property is important for the API layer (and application logic) to follow the state transitions within the framework correctly. For instance, the progress states of a connection between two devices conform to a chronological sequence of events from starting a connection, up to performing a handshake, and eventually being in the connected state. In order to preserve the sequence of notifications, the notification-broker runs only one thread, which delivers the enqueued notifications one after another to the API layer. In turn, the API layer (and application logic) must not block the notification thread. Otherwise, the flow of notifications and in consequence the functionality of asynchronous calls would get stuck.

Sequence order
guarantee of
notification-broker.

As notifications on some platforms have to cross a computationally expensive border, the notification-broker mostly communicates through a few numeric values and bitfields with the API layer, for example, to identify the actual notification, a certain state machine, or files. Data is only attached to a notification if its size is very small, such as short text messages. Large amounts of data, such as files, have to be loaded on demand by application logic. This kind of communication enables a maximum in notification throughput and speed and is most im-

Performance of
notification-broker.

Platform dependent
callbacks of
notification-broker.

portant for platforms that are based on high-level run-time environments for application development. For instance, notifications on Google Android platforms have to cross the border from native code to the Java virtual machine through JNI¹⁶. Other examples are .NET-based platforms where communication has to cross the border from so-called unmanaged code (native) to the managed .NET run-time. Such high-level run-time environments employ their own memory and resource management, which are usually characterized by garbage collections. Hence, those run-time environments cannot simply access memory and resources from native code without additional platform dependent logic. However, simple data types such as numeric values or bitfields can be directly exchanged without further consideration.

5.4.6.3 Mediator Layer

Responsibilities of
mediator layer.

An essential task in realizing distributed MSE applications is to establish and maintain connectivity between corresponding application instances. Several sub-tasks belong to this such as detection of corresponding application instances and determining its associated (network) connectivity details, establishing the actual connections on demand, keeping the connections alive, and gracefully closing the connections as well as releasing all resources that have been acquired during the whole interaction sessions. What makes this task more complex is the fact that application instances within MSEs may appear, vanish, or reappear in an ad-hoc manner, which requires appropriate consideration in error-handling and maintaining of the session states.

Mediator layer as an
approach to tackle
challenge 3.

The mediator layer together with the network layer is responsible for those tasks and addresses the *challenge 3* of ► Subsection 5.4.3. Both layers work tightly together and code distribution of one layer on a few places goes fluently into the other layer. However, the functionality of both layers can be clearly distinguished. Hence, we discuss the functionality in the following separate sections.

Mediator layers of
applications talk with
each other.

The mediator layer is responsible for management of devices (and application instances) that are available within MSEs. Each application instance's mediator layer "talks" to the mediator layer of other application instances in order to mediate connectivity details for the network layer. As a result of this task, it knows about all currently available application instances and maintains a list of them, which is served to application logic as well as other framework layers on demand.

¹⁶Java Native Interface

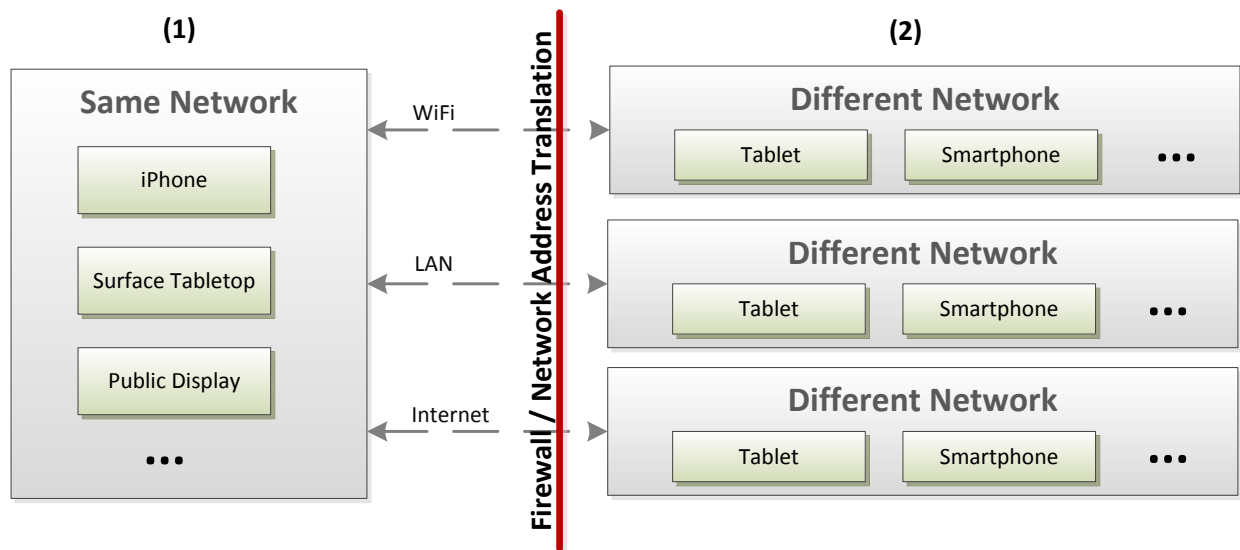


Figure 5.17: Mediator network scenarios: (1) Devices that are in the same network. (2) Devices that are in different logical or physical networks.

This list covers application instances for the following two scenarios as sketched in ► Figure 5.17: (1) devices that are in the same network and are able to contact each other directly; (2) devices that are located in logically or physically separated networks and are not directly accessible. Devices in different networks are typically separated by at least two (but usually multiple) firewalls and gateways, which make it difficult to contact each other directly. For example, scenarios where mobile devices are connected to a wireless private network provided by a NAT¹⁷ router whilst corresponding large static displays are connected to a different network provided by a LAN. Such scenarios often occur in research labs where multiple wireless and wired private networks for dedicated projects or applications coexist.

DEVICES IN THE SAME NETWORK. The mediator component in each application instance employs broadcast messages to maintain a list of all available application instances within the same network. Broadcast messages are received by every device on the same network. Hence, they suit well to exchange application instance identifiers with each other. An application instance identifier contains the application ID, the area / application name (cf. ► Subsection 5.4.4.5) and connectivity details. Upon start of the mediator layer instance, a greet message is broadcast to the network to tell other instances about the existence and availability of the application instance. Every mediator instance that receives a greet mes-

Device management by maintaining two lists.

Devices in the same network make use of broadcast communication.

Main broadcast communication.

¹⁷Network Address Translation

Device ID assignment.

sage broadcasts its own application identifier to inform the availability of itself, thus update its alive-status to other mediator instances. On exit of the mediator instance, a bye message is broadcast to the network in order to tell other mediator instances about the absence of itself. These broadcast messages are also used to determine and negotiate application instance IDs as mentioned in ► Subsection 5.4.4.5. For such IDs, mediation is required in case of conflicting usage of IDs, where the same numeric ID is used by multiple application instances at the same time. If such a case occurs, the mediator layer instances negotiate the application instance IDs with each other and automatically re-configure their mediator layer if necessary.

Devices in different networks employ mediator server instance.

DEVICES IN DIFFERENT NETWORKS. In order to support environments where the underlying network structure of an MSE is composed of multiple private networks, which have no direct route to each other but have network routes to external addresses through NAT, the mediator layer builds on a mediator server instance, which all devices have access to. This setup is optional and only required for connections across different networks. Mediator layers of applications can register at a mediator server instance, which maintains a list of registered application instances with their application instance details. This list is retrieved by the application's mediator layers and augments the list of available application instances (within the same network) with those not directly available (from a mediator server). This enables applications to determine application instances within a logical MSE even if the devices reside in completely different networks. Furthermore, the mediator server instance helps application instances connect each other across different networks by means of the mechanisms STUN¹⁸ for TCP and STUN¹⁹ for UDP.

Mediator server instances help establish direct end-to-end connections.

5.4.6.4 Network Layer

Responsibilities of the network layer.

The network layer is responsible for establishing connections to other devices and transferring messages and files or data buffers between devices. It is designed to support interactive systems through selectively distributing the data to be sent to appropriate transport channels. Of utmost importance for the responsiveness of interactive applications is the rule that the communication of custom application logic with other application instances (e.g., status updates, commands, or requests) must not be delayed as best as possible. Furthermore, real-time data such as touch events or video stream packets have to be transferred as fast as possible without affecting the application logic's communication. Therefore, the network layer operates with different transport channels as described in the following section. In preparation of establishing connections, the network

Network layer maintains three different channels.

¹⁸RFC5382 <http://tools.ietf.org/search/rfc5382>

¹⁹RFC5392 <http://tools.ietf.org/search/rfc5389>

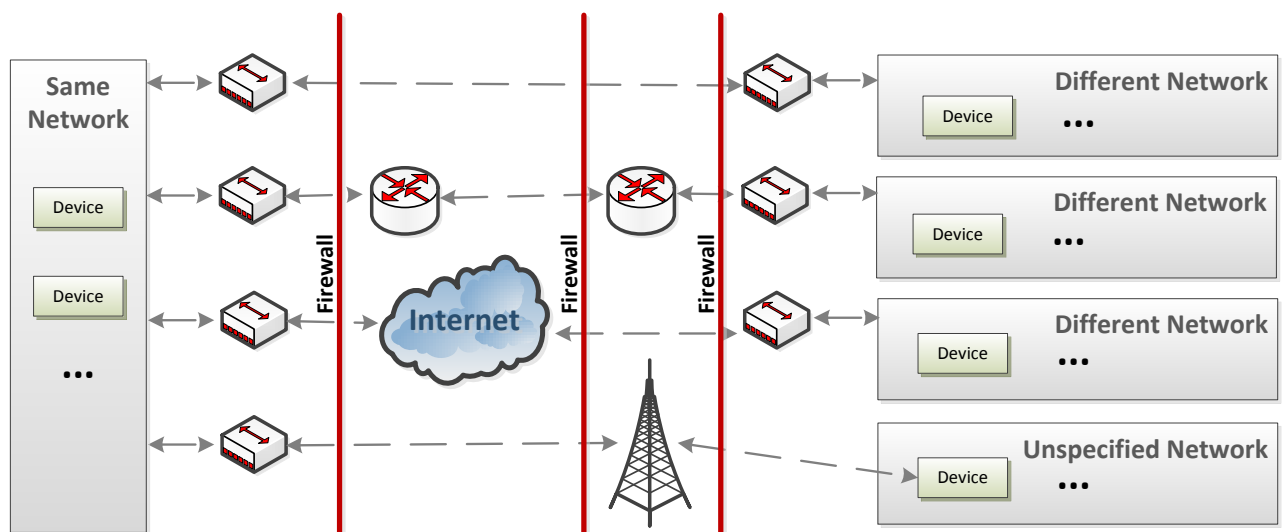


Figure 5.18: Network scenarios supported by the Environs framework.

layer interacts with the mediator layer to retrieve connection details for the mediator scenario (1) or to employ the mediator server instance's service to initiate STUNT/STUN channels in case of the mediator scenario (2). STUNT/STUN channels are established through TCP/UDP hole punching as published in detail by Ford et al. [2005] for peer-to-peer communication.

CONNECTIONS BETWEEN DEVICES. Upon successful connection with a device, the network layer has established the following channels: (1) TCP communication and data channel; (2) TCP interactive channel; (3) UDP interactive channel. The first channel (1) serves as the communication channel for custom application logic as well as framework communication with other framework instances, for example, to start/stop a video portal or handshake options for the video portal. Large files or data buffers area also handled through the channel (1). However, the transmission of such data potentially takes more time depending on the size of the data and would induce lag and wait times to communication of application and framework logic. Consider, for example, the transmission of a file that is larger than 20 MB from one device to another one over a low bandwidth route such as a mobile network, which would take several minutes to complete. This transmission would block and delay all other transmissions for that amount of time (cf. diagram (1) in ► Figure 5.19). Therefore, such transfers are split by the framework into multiple smaller units and transported one after another in order to allow other messages to be transmitted between the units (cf. diagram (2) in ► Figure 5.19). This mechanism in combination with the assignment of higher priorities to application and framework communication enables responsiveness of interactive applications.

Transfer of large amounts of data over communication and data channel (1) is performed in many smaller transmissions.

Prioritization of data on the channel (1).

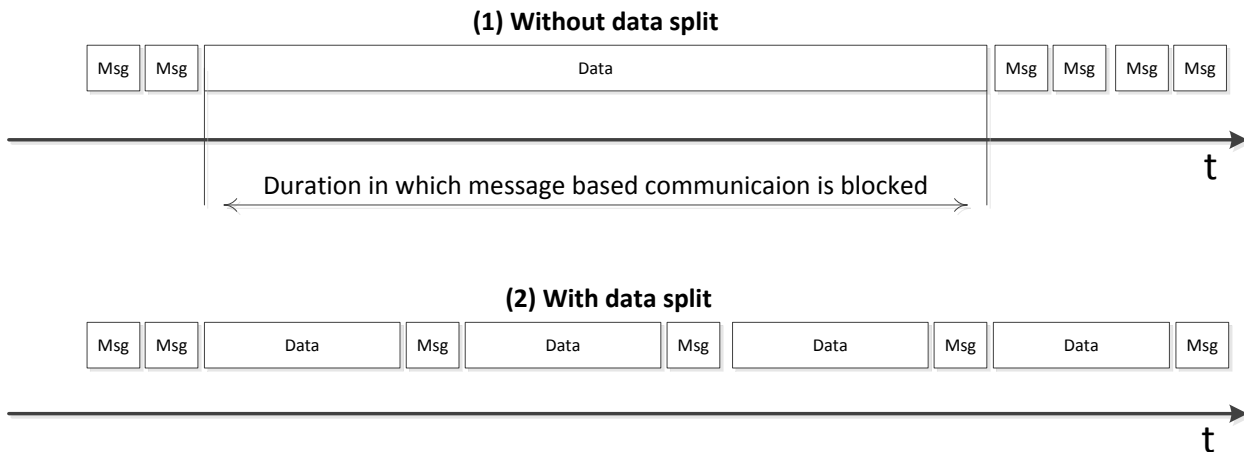


Figure 5.19: Split of large data for transport over the communication channel. (1) Progress without split. (2) Progress with split.

Interactive channel (2)
low latency data.

Interactive channel (2)
used on demand.

The interactive channel (2) is employed only on demand for data that is related to user interaction or interface visualization. In comparison to the channel (1), which is tuned for optimal bandwidth throughput through the channel, the options of the channel (2) are tuned in terms of socket buffer sizes and behavior to achieve as low latency as possible. Bandwidth throughput of the channel (2) might not be optimal, but lags due to network related protocol algorithms are reduced as best as possible allowing packets to be transferred with the lowest latency that can be achieved. In addition, while channel (1) is an always active channel, channel (2) is designed to be inactive unless interaction related tasks are active and ongoing. Hence, resources (e.g., buffers, threads) for the interactive channel are allocated only if interactive portals need to be streamed or user interaction related data need to be reliably transmitted. As soon as interaction related tasks stop, all resources for the channel (2) are released (unless those required to keep the connection alive) and the channel is put back into a sleep state.

UDP interactive
channel (3) for
unencrypted low
latency data.

Drawbacks of UDP
interactive channel
(3).

The UDP interactive channel (3) is dedicated to touch events and sensor data due to the time constraints of such kind of data in order to achieve as best response time as possible for interaction and user interface related tasks. UDP packets are transported faster than TCP packets over networks as they are not withheld or buffered by any underlying network stacks for reassembling, retransmission requests or reordering due to sequence order guarantees. However, such low latency transport comes at the expense of potentially dropped packets, that is, UDP packets are not guaranteed to arrive at the destination at all. Furthermore, the sequence in which packets are sent is not necessarily the same sequence in which packets arrive at the destination. All packets may take different routes and some of them may be faster than others and arrive earlier at the destination.

In spite of the drawbacks, UDP transport suits well for touch events and sensor data if potentially dropped packets and disordered arrival of packets are taken care of by appropriate mechanisms. For example, users would not notice missing intermediate touch events during a touch gesture or missing intermediate compass values when rotating a tablet. However, they severely notice the lag in the visualization of the effect of such events. If touch events are delayed multiple times due to retransmissions of lost touch events, then their occurrences as well as according visualizations are shown timely disrupted on the destination with potentially negative effect on user experience.

Trade-off for UDP
interactive channel
(3).

HANDSHAKE. Upon successful connection of the main channel (1), the devices exchange their capabilities, such as device type (tablet, smartphone, tabletop, display, etc.), screen dimensions in pixels, display density in DPI, support for video formats, availability of sensors, or channel/network details. Those capabilities are automatically detected by the framework and serve to optimize the transport channels or to automatically derive parameters for interaction and visualization. For instance, if custom application logic has not specified the size of a video portal on a tabletop device, then the video portal's size is calculated to match the area that the device covers on the tabletop surface by means of pixel and DPI values.

Basic application and
device identification.

SCALABILITY. The network layer is designed to be highly scalable in terms of the number of manageable devices within an MSE and communication with them. Communication between devices through short messages does not require the involved devices to be connected together. The network layer automatically determines the best route between two devices (e.g., direct access through end-to-end channels between devices or through a mediator server) and handles message exchange through the best option where it always prefers the fastest route.

Scalability in terms of
the number of devices
at the same time.

Interactive portals as well as interaction through portals and exchange of large files and buffers require devices to connect to each other at first. Such a connection requires resources such as memory buffers, threads, or encoder/decoder. As mobile surfaces are limited in resources available to applications, the framework limits the maximum amount of device connections depending on available resources.

Scalability in terms of
resources.

5.4.6.5 Touch/Sensor Layer

Instant visual feedback and quick response to user input are important properties of interactive applications with a strong impact on user experience. User demands and expectations on such properties have increased quickly in the recent past with every new generation of devices that were released with every

year. Hence, it is more than likely that demands on those properties will also be claimed for applications such as envisioned for MSEs.

Responsibilities of touch/sensor layer.

Native access to touch/sensor data.

RESPONSIBILITIES. The touch/sensor layers in application instances address those properties (to fulfill requirements 2 and 4 in ► Subsection 5.4.3) with three responsibilities: recording of events, putting them through transport channels (cf. ► Subsection 5.4.6.4) to devices that have registered for them, and processing them at the target device. All responsibilities must be performed as fast as possible in order to achieve low response times. Therefore, recording of events happens in native code using native APIs whenever possible. For example, on Apple iOS or OSX platforms, the layers employ native access to touch events and sensor data through *CoreLocation* and *CoreMotion* of the Apple platforms. On Microsoft Windows platforms, the layers access the native *LocationAPI* through COM-objects²⁰ to obtain GPS locations. However, there are cases where native access is not allowed or concealed. In such cases, the touch/sensor layers obtain touch/sensor data with the aid of the platform layer by means of platform-specific public APIs. For instance, on Microsoft PixelSense tabletops, the platform layer receives touch events from the NET-runtime and puts them through to the native touch/sensor layer for further processing.

Low latency, consistency, and safety of sensor data transport.

The second responsibility of this layer is twofold. First, event data generated from input sensors have to be transported from a source device to destination devices as fast as possible. Second, event states must be kept consistent at the destination device in case of packet drops or packet sequence disorders. Environs provides two channels for conducting event data to destination devices: (1) a secure and lossless TCP channel and (2) a fast but unsecured UDP channel. By default, Environs classifies event data to be either security sensitive (for channel 1) or security uncritical (for channel 2) for conducting through the according channel. However, applications can choose the desired channel for each type of event data individually if default channel settings would not be appropriate.

Security sensitive sensor data, e.g., GPS location or heart-rate.

Trade-off between latency, safety, and security.

LATENCY AND SECURITY. Security sensitive data are those that would expose private or critical user related data if transported in plain format over network channels. For example, GPS locations are considered security sensitive as such data would enable anyone – who is able to intercept the channel – to determine and track a user’s absolute location. Another example is the heart-rate of a user (e.g., measured by a smartwatch), which might be reasonably employed to adapt interaction techniques but may expose health-related inferences to interceptors. Therefore, security sensitive data is conducted through the encrypted TCP channel 1. The drawback of channel 1 is a little higher latency of at least a few milliseconds due to TCP transport’s congestion control²¹ and additional encryption/decryption steps. However, the additional latency is uncritical for most

²⁰Component Object Model

²¹RFC 5681 <https://tools.ietf.org/html/rfc5681>

heart-rate or GPS location-based applications as users would not notice whether the visualization of positions is delayed for a few milliseconds or not.

Security uncritical data are those that do not expose private or personal user data. For example, data from accelerometer sensors are considered security uncritical as they only provide relative changes detected by the sensors. Without further knowledge of their actual usage in applications, such kind of data exposes only a little information. However, such sensor data needs to be transported as fast as possible if they serve for enabling immediate and responsive user interaction. Therefore, security uncritical data is conducted through the unencrypted UDP channel 3.

Security uncritical sensor data, e.g., accelerometer or gyroscope.

As discussed in ► Subsection 5.4.6.4, the UDP channel is extremely fast at the expense of packet drops and packet sequence disorders. In order to compensate for such drawbacks, the touch/sensor layer assigns an incremental sequence number to each data packet. By this means, belated packets can be detected and dropped and the correct sequence of packets can be recognized and guaranteed. This mechanism is similar to the one employed in the TUIO protocol by Kaltenbrunner [2009]. As packets can be dropped – caused by network behavior as well as due to sequence order guarantees – Environs applies a further compensation approach, which derives and interpolates missing states and events (if appropriate) to keep touch and sensor data states consistent. For example, if a touch frame (encapsulated into a data packet) indicates that some of the touch contacts (that existed before) are not available anymore but the according events have not been seen, then touch up events are generated and inserted in order to keep according state machines consistent.

Compensation approach for packet drops and consistency of state machines.

GESTURE RECOGNIZER ARCHITECTURE. In addition to the compensation approach, the touch/sensor layer implements a versatile gesture recognizer chain architecture, as sketched in ► Figure 5.20, that is based on plug-ins provided by developers. Multiple gesture recognizer plug-ins can be attached to the sensor data stream of a sensor at the source device as well as at the destination device before the sensor data is dispatched to application code. Each plug-in in the chain is fed with raw sensor data one after another as indicated by the black colored continuous lines. Each plug-in is able to control whether subsequent plug-ins in the chain shall be invoked or not as well as whether sensor data of the stream shall be sent to destination devices. This gesture control mechanism (as indicated by the brown colored dashed lines) is useful for plug-ins that need to take control over the subsequent sensor data flow in order to realize interaction techniques. For example, if a plug-in has recognized a gesture for zooming or pinching, then destination devices are notified through gesture commands with appropriate arguments for zoom position and current zoom level. Gesture execution will go on until the recognizer detects the end of the gesture. If at the same time, touch contacts are visualized on the destination device or subsequent recognizer plug-

Gesture recognizer plug-in chain.

Recognizer chain control mechanism.

Hold off touch events during (zoom/pinch) gestures.

SUPPORTED SENSORS. The current development state of Environs is prepared for a wide range of sensor types that are commonly available in mobile tablets, smartphones, smartwatches, and laptops (i.e., Accelerometer, gravity, gyroscope, magnetic field (compass), device orientation, ambient light, proximity sensor, GPS (location, speed, and altitude), altimeter (air pressure), rotation, heart-rate, VOC, humidity, temperature). However, Environs is not limited to the supported sensors but enable custom sensor plug-ins to be integrated into the sensor layer as data source.

Support for common mobile device sensors.

5.4.6.6 Interactive Portal Layer for Smart Portal

In order to enable interactive video portals, Environs contains a dedicated interactive portal layer that realizes so-called *Smart Portals*. Such portals are designed to provide high-quality, high-resolution video streams optimized for low latency. *Smart Portals* replicate part of a source window, such as the application visualization of an interactive tabletop, to the application window of another device. The interactive portal layer automatically renders the video stream to the window background of an application window specified by the user application.

Responsibilities of the interactive portal layer.

The interactive video portals are called *smart* mainly because of three reasons. First, the Environs framework automatically detects and employs the most appropriate video parameters (e.g., video resolution from the available and supported resolutions of both devices) with low latency and optimal network transport as the primary goal.

Why are portals smart?

1. Smart parameter selection.

Second, the interactive video portals are orientation-aware as exemplarily sketched in ► Figure 5.21, that is, the video's source may be rotated by any degree and rotation may change at any time through users, which perfectly suits situations at tabletops where users may be located on every side of the device. In contrast, most interactive video portal applications require the video's source to be a rectangle that is orthogonally aligned to the x/y-axes.

2. Orientation-awareness.

Third, Smart Portals automatically conducts touch interaction with the target device back to the source device and injects the touch events exactly at the portal destination's locations. That is, the coordinates of the touch events are transformed into the coordinate system of the source device's workspace coordinate system, which includes rotation (for rotated portals), translation, and scaling.

3. Smart touch support.

Smart Portals enable developers to easily build interactive portal applications as regular applications taking advantage of operating system widgets without the inclusion of additional external stand-alone applications. The following design elements of the interactive portal layer are decisive contributions to achieve low latency and high-resolution portals:

Decisive design elements of Smart Portals.

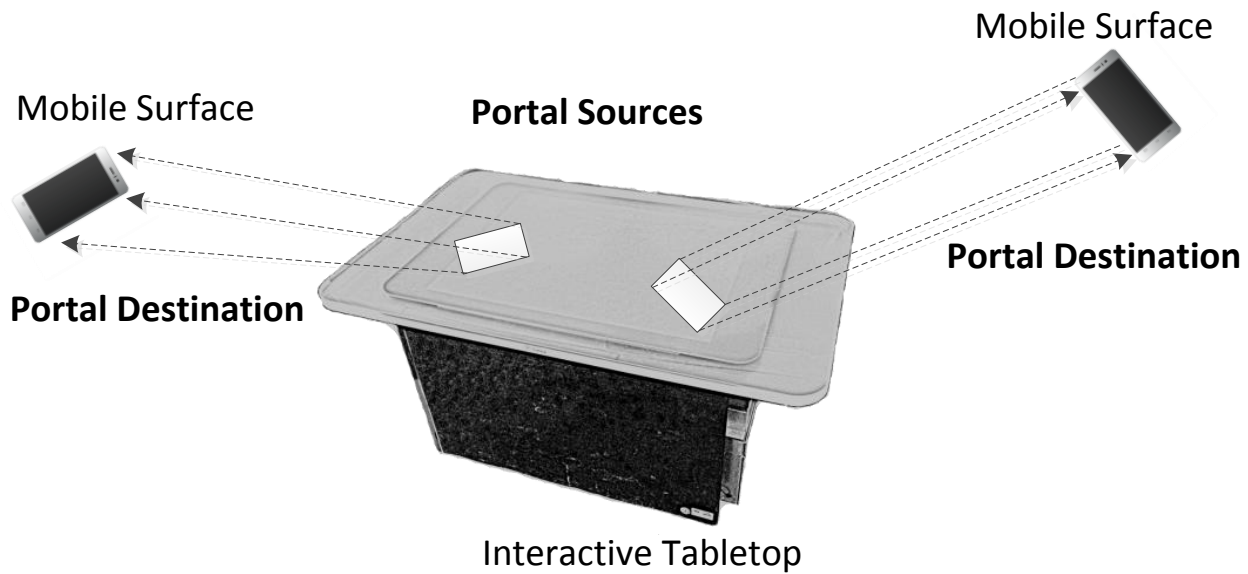


Figure 5.21: Orientation-awareness of interactive portals. Portal sources on the tabletop surface may be rotated by any degree.

Hardware supported
video compression.

1. *Video compression* is used to minimize latency induced by network transport. *Smart Portal* employs the high-efficiency video codec H.264 enabled through cisco's open-source implementation OpenH264^a based on the lowest latency encoding profile.

This library is available as native code for all platforms and used for encoding as well as decoding. However, software decoding through OpenH264 is only used as a fallback case. The framework makes use of encoding and decoding by dedicated hardware if available, which unburdens the CPU from video encoding/decoding while application and framework logic fully benefits from the CPU.

Hardware supported
rendering.

2. *GPU acceleration:* Virtually all nowadays' graphic cards support scientific computation by means of the standardized OpenCL^b API. For this reason, computational intensive preprocessing steps of the video stream's source images are performed on the GPU for which we developed optimized OpenCL kernels.

^a <http://www.openh264.org>

^b <http://www.khronos.org/opencl>

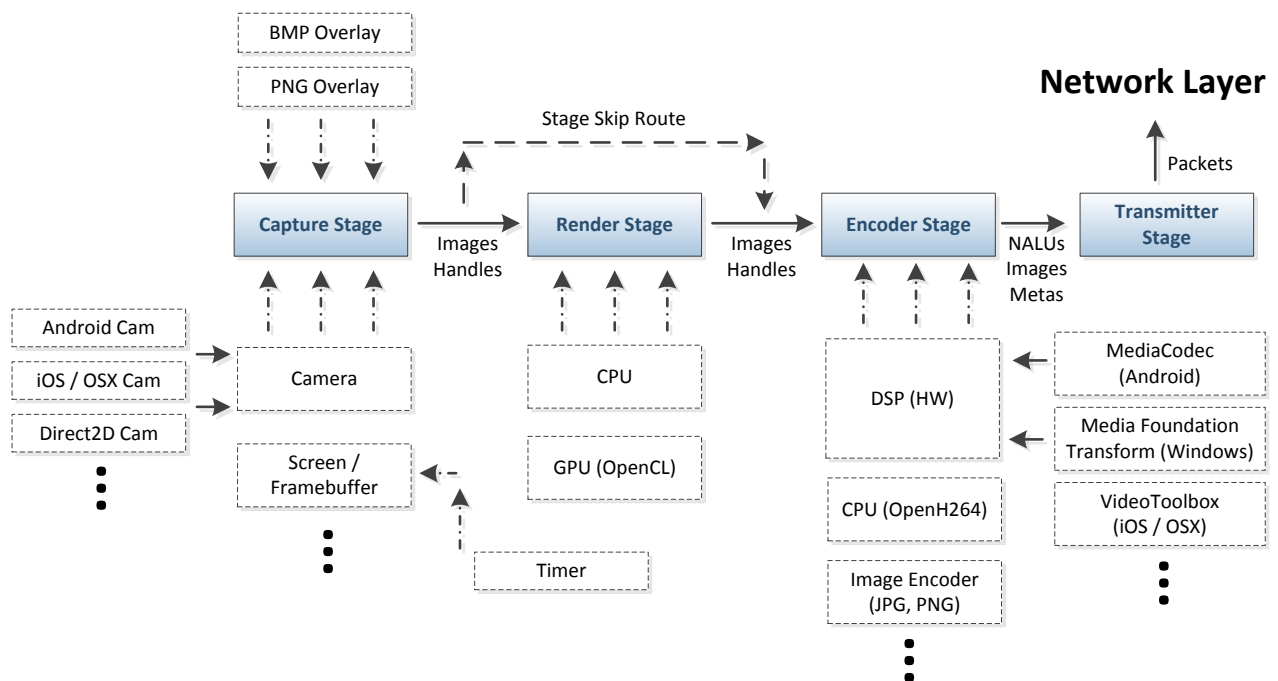


Figure 5.22: Portal creation pipeline stages and stage plug-in types of the interactive portal layer.

Both elements are widely supported on modern mobile and static surface platforms, which enable compatibility of smart portals between many different device platforms in a heterogeneous device ecology. However, if a platform does not fully support both, then the Environs framework provides fallback options, which even very old devices do support, that is, employing the CPU to enable video portals by means of a sequence of JPEG images.

Compatibility and fallback mechanisms.

PORTAL CREATION AND VISUALIZATION PIPELINES. The biggest challenge in designing the interactive portal layer was to devise a concept that enables efficient and low latency processing of streams while at the same time being flexible enough to support many different video portal combinations, configurations, and platforms. For example, the same layer implementation must support detection of and access to available hardware for encoding, decoding, and rendering of video images as well as supported configurations (e.g., resolutions, color spaces, or image encodings) for many diverse media APIs (e.g., Android MediaCodec, iOS/OSX VideoToolbox, Windows Media Foundation Transforms) with different programming paradigms. Due to so many differences, the interactive portal layer separates portal creation and visualization into flexible pipelines with several stages, where each stage is created and configured depending on the actually available platform support at run-time.

How to maximize flexibility and performance of portal creation.

Approach of portal creation and visualization.

Pipeline stages and optimal throughput.

PIPELINES TOGETHER WITH PIPELINE STAGES. are often used concepts in software and hardware design to realize separable steps of a process, which depend on one another in a sequential order. Pipelines enable to maximize parallelism by means of multiple worker-threads that concurrently drain the stages as best as possible. For example, the capture stage in ► Figure 5.22 continuously captures images one after another and submits them to the next stage instead of waiting until a captured image has passed through all pipeline stages. Thus, the pipeline architecture increases throughput and reduce latency if appropriately configured. Even on single core platforms that can execute only one thread at any time, this architecture increases throughput, as each stage takes some time (e.g., encoding through hardware, rendering on the GPU) to finish, during which the main processor is scheduled to another thread. Here, the pipeline architecture enables the one single core to advance to other stages while one stage is busy. We will next discuss the portal creation pipeline as sketched in ► Figure 5.22 and afterwards the portal visualization pipeline in ► Figure 5.26.

Reusing of the pipeline for all platforms.

PORTAL CREATION PIPELINE. The portal creation pipeline (cf. ► Figure 5.22) consists of four stages whereof three stages are realized as dynamically replaceable plug-ins. By this way, the architecture enables using the same base layer implementation for all platforms. Platform-specific access to, and run-time availability of, appropriate hardware, as well as platform-specific fallback mechanisms, are provided by dedicated native plug-in implementations. As long as stage plug-ins conform to their according stage interfaces (C++), each implementation of a stage plug-in can be dynamically loaded, probed, and instantiated at run-time and finally placed into the pipeline. A substantial part of the probing-process involves negotiation of an optimal configuration between each pipeline stage from a set of available and supported options. For this negotiation mechanism, the pipeline implements a forward-propagation approach, which starts at the capture stage. Each stage talks with the next stage in order to mediate the best configuration with the aim of low latency processing.

Pipeline stage negotiations for determining an optimal configuration.

Fallback stage implementations.

Due to the dynamic behavior of the pipeline initialization and creation, the process to set up a working pipeline may also fail. First of all, availability of hardware support on target platforms is quite likely to be present but not guaranteed. Very old devices or misconfiguration of a system may result in the lack of hardware support. Another reason is concerned with the available resources at run-time, which may vary depending on other activities of the same application or other applications running on the same device at the same time. For example, if multiple portals are already active, then establishing a further portal pipeline might fail if the number of concurrent encoding/decoding processes would exceed the limitations of the hardware. Therefore, both pipelines (creation and visualization) provide fallback mechanisms for interactive portals if all available plug-ins fail. Such fallback mechanisms are implemented for all platforms and

guaranteed to work in any case and establish interactive portals based on sequences of JPEG or PNG images. Image sequences require more (with JPEG) respectively much more (with PNG) network bandwidth than video codec based streams. However, if sufficient network bandwidth is available, then the performance of image streams is comparable with the performance of the well known VNC screen sharing.

CAPTURE STAGE. The first stage in the portal creation pipeline is responsible for capturing images of the portal source and submitting them to the next stage as fast as possible.

Portal sources can be of different types as depicted in ► Figure 5.23: video cameras, application windows, or frame-buffers. Depending on the particular platform, capture stage implementations always try to capture with the fastest available approach, for example, direct access to WPF window buffers or Direct3D buffers on Windows platforms. ► Figure 5.23 also shows that triggering of the capture stage, that is signaling the stage to capture an image at a given frame rate, depends on the type of the capture source. Cameras usually have their own trigger source, which is determined by supported and configured frame rates. The other types of capture sources create an instance of a pipeline timer that triggers the capture stage at a rate of 30 frames per second by default.

Before submitting a frame to the next stage, the capture stage can be configured to superimpose the frame with multiple image overlays, which exploits the transparency or alpha channel of the overlay images. Image overlays can be changed by application code at any time if necessary (e.g., on overlay content change) and enable augmentation of the interactive portal with additional information in real-time. One example application for image overlays is given in Dang et al. [2015] in which we demonstrated the usefulness of image overlays in the context of group visualizations. We realized a distributed application that employed interactive portals to show energy consumption diagrams on a tabletop device as well as on mobile tablet devices. Image overlays were used to augment the interactive portals with cutlines and explanations depending on the actual type of diagram visualizations.

RENDER STAGE. The second stage in the portal creation pipeline is the render stage (cf. excerpt in ► Figure 5.24), which is responsible for transforming images of the previous stage for the configured portal destination. The stage tasks include the following image processing operations:

- **Comparison of subsequent frames** is a quite important step and helps reduce the system load by skipping all remaining image processing steps in case of equality of a frame with the frame of a previous cycle. Usually, this happens quite often within a portal image stream. If the portal source

Performance of fallback stage.

Responsibilities of capture stage.

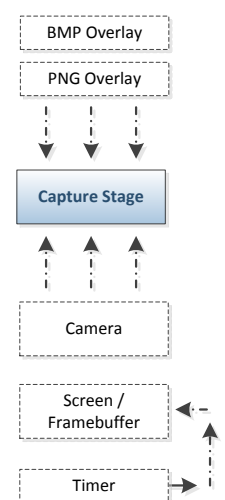


Figure 5.23:
Capture stage.

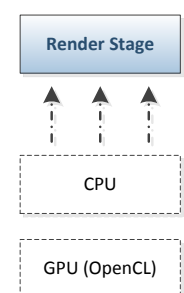


Figure 5.24:
Render stage.

Skip redundant (but intensive) processing.

has not changed, then the render stage either reroute the pipeline to the last stage or just reuse the cached result of the last run depending on the requirements of the particular combination of encoder stage and decoder stage.

The aim of this task in the best case is to skip almost everything in the pipeline and just tell the portal destination with a small message of a few bytes in size that nothing has changed instead of performing the same operations as in the previous cycle, encode the same frame as in the previous cycle and sending the result to the portal destination. However, some encoders require a full image frame at the configured frame rate and some decoders require stream input to have a constant load in order to function correctly, for example, to guarantee a configured presentation time. If one of the components has such a requirement (the worst case), the render stage just supply the encoder stage with the cached frame from a previous cycle instead of performing all the image processing operations again.

Support portal
resizing.

- **Bilinear scaling** is required since interactive portals support arbitrary portal source sizes, which may be changed by the user at any time while the portal is active. Hence, each captured image needs to be scaled to the target video stream resolution of the encoder stage.

Orientation
awareness.

- **Rotation by a given angle** is a requirement for tabletop surfaces since mobile devices may be placed arbitrarily oriented on the tabletop as shown in ► Figure 5.21.
- Finally, **format and color space conversion** are usually requirements for video-encoding processes as those often do not support bitmap formats (e.g., RGB) but video formats (e.g., YUV).

Hardware supported
rendering.

REAL-TIME GPU PIPELINE. As sketched in ► Figure 5.22, all the image processing tasks can be performed by the CPU on every platform. However, most of the platforms feature modern and highly powerful GPUs and those GPUs have only little workload most of the time caused by rendering the operating system and application user interface. Furthermore, modern GPUs have a multitude of independent computing units where each unit can run a multitude of work items ($>> 256$) in parallel and extremely fast, thus process much more pixels of an image in parallel and much faster than the CPU can do. Therefore, in order to increase performance and reduce the latency of interactive portals, the framework provides a GPU render stage plug-in, which performs all the render steps on the GPU based on the standardized OpenCL API. Thereby, the render stage unburdens the CPU, which results in preserved computing resources for the benefit of application and framework logic, which further reduce system latency.

ENCODER STAGE. The encoder stage (cf. excerpt in ► Figure 5.25) takes over images from the previous stage and encodes the images into a stream format that was requested by the portal destination, which can be an H.264 video stream or a sequence of JPEG/PNG images. The encoder stage also supports H.265 aka HEVC²² streams, which offer much better encoding efficiency. However, since this codec is relatively new, only recent devices are able to encode and decode such streams. In order to provide compatibility with as many platforms as possible, the encoder stage relies on the H.264 codec and makes use of H.265 only on explicit request by applications.

Whenever possible, the pipeline probing-process during pipeline creation tries to load and establish a hardware-accelerated encoder plug-in. Hardware-accelerated encoders are available on almost every modern platform either as an integral part of the CPU, a CPU/GPU extension, or a dedicated component on a SoC²³. Such encoders enable multiple encoding tasks at the same time, for example, to enable video conference calls with more than two participants, which require at least one encoding task for each participant of the call. Overall, hardware-accelerated encoding reduces the latency of interactive portals and un-burdens the CPU a lot. Furthermore, even small mobile devices are capable of multiple interactive portals at the same time while the CPU still has much computing resources left for executing application code. If hardware-accelerated encoding cannot be established, the encoder stage falls back to software encoding using the open-source implementation OpenH264 of Cisco Systems. Software encoding, however, burdens the CPU with each interactive portal, thus limits the amount of interactive portals that run at the same time.

TRANSMIT STAGE. The last stage in the portal creation pipeline is responsible for transmitting the stream packets to the destination device by means of the network layer. For each interactive portal, the transmit stage activates the TCP interactive channel (2) between two devices (cf. ► Subsection 5.4.6.4) and sends the stream packages as fast as possible through the channel. As soon as the interactive portal stops, the transmit stage deactivates the interactive channel in order to release as many resources as possible.

PORTAL VISUALIZATION PIPELINE. Each interactive portal ends at a portal destination where a portal visualization pipeline (cf. ► Figure 5.26) is fed with portal packages received by the network layer. The portal visualization stages basically perform the reverse operations of the corresponding stages in the creation pipeline. Portal packages are handled first by the receiver stage, which mainly buffers the received packages and carries the buffers over to the decoder stage one after another.

Responsibilities of encoder stage.

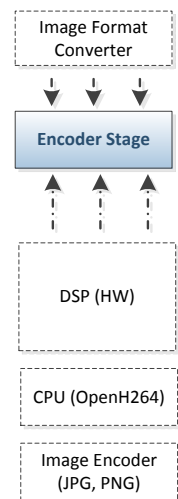


Figure 5.25:
Encoder stage.

Fallback for encoding.

²²High Efficiency Video Coding

²³System on a Chip

Network Layer

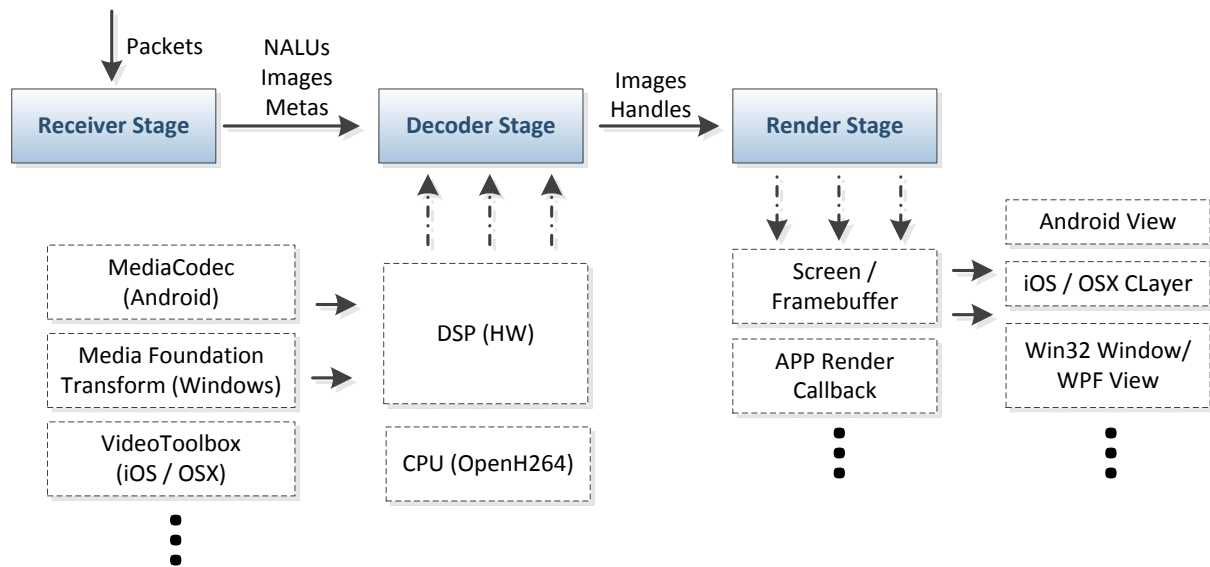


Figure 5.26: Portal consumer/visualization stages of the interactive portal layer.

Transform streams to images.

DECODER STAGE. The decoder stage works similar to the encoder stage in the portal creation pipeline and also prefers hardware-accelerated decoding. In contrast to the encoding stage, the decoding stage decodes the video stream or image sequence to bitmap images or platform-specific image handles for the render stage.

Visualize image sequences for applications.

RENDER STAGE. Finally, the render stage is responsible for presenting each decoded bitmap image to users in order to visualize the interactive portal. Visualization of interactive portals can be performed by two mechanisms. The first one is the most comfortable method where applications just have to provide a platform-specific handle to a destination for visualization. For example, Google Android platforms have to provide an Android View or Surface, Apple iOS or OSX platforms have to provide a CLayer, and Microsoft Windows may provide a WPF Window handle or a Win32 Window handle. The render stage then exploits the approach with the best performance to render decoded bitmaps onto the provided visualization destination.

Applications may take over visualization.

The second approach makes use of a callback method provided by application code. For each decoded bitmap, this callback method is then invoked with direct memory access to the bitmap data as a method parameter. Thus, the second approach enables applications to render portal visualizations by using application-defined approaches.

LATENCY MEASURES. In order to show the efficiency and performance of the portal implementation, we measured the latencies by means of a simple technique in which we superimposed the portal's source with a number that increased with every frame at 30 fps. This number is then replicated through a portal to a tablet device. Both devices are photographed together with the increasing number as depicted in ► Figure 5.27. Based on the difference of the numbers and



Figure 5.27: (a) Latency measure of a portal that covers the tablet's physical size on the surface;
(b) Latency measure of a high resolution portal source.

the frame rate of 30 fps, we determined the latencies shown in ► Table 5.2 for a Microsoft PixelSense 2.0 device and a Samsung Galaxy Tab 2.

Video stream size	Min. (ms)	Median (ms)	Max. (ms)
294 x 454	66.6	99.9	133.2
844 x 1080	99.9	133.2	166.5

Table 5.2: Latencies at 30 frames per second ($\pm 16ms$).

► Table 5.2 lists the latencies of a TCP portal for two video stream resolutions, that is, 294x454 (pixels of the surface covered by the tablet's physical size, cf. ► Figure 5.27a) and 844x1080 (full height of the surface tabletop, cf. ► Figure 5.27b). For the measures in each row of the table, we took at least 30 pictures in sequence and determined the median, the lowest, and the highest latency. The average latencies are between 100ms and 133ms, which we consider low for such a complex MSE system. The difference between the video stream sizes are 33.3ms on average, which gives a strong indication that the main part of the latency was induced by network transport. Smaller video stream resolutions yield fewer video data to be transmitted, which in turn can be displayed earlier on the portal destination. Therefore, the results revealed possibilities for further latency improvements through network optimizations.

5.4.6.7 Encryption Layer

Responsibilities of encryption layer.

Secure communication over unsecured networks became more and more important during recent years as the proliferation of network linked mobile devices as well as mobile application usage rapidly increase with every year. The only way to guarantee privacy and safety of personal and private data is to secure communication with cryptographic methods, which are meanwhile well known, standardized and available on every platform. It is more than likely that multi-surface frameworks must provide means to enable secure communication. The Environs encryption layer addresses this issue and basically creates a secure hull around communication channels between devices. Every byte that goes through such a secured channel is encrypted and therefore safe against interception attacks.

Encryption and heterogeneity of platforms.

The design of Environs' encryption layer shows how to realize secure communication at framework level through the lens of the challenges and requirements described in ► Subsection 5.4.3. In particular, supporting heterogeneous device platforms introduces considerable complexity into the application of cryptographic functionality due to different platform-specific cryptography APIs with different access mechanisms, multi-threaded sync and lock handling, data formats for certificates or keys, or endianness²⁴. Therefore, the crypt layer specifies a unified API to access cryptographic mechanisms, which works exactly the same on every supported platform.

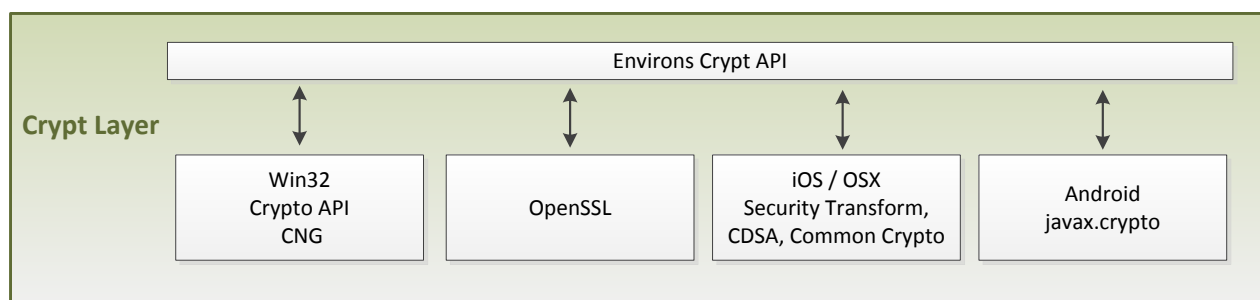


Figure 5.28: Environs Crypt Layer.

A unified Environs crypt API.

A rough sketch of the crypt layer composition is given in ► Figure 5.28. Access to cryptographic mechanisms (by another framework layer) is provided through the *Environs Crypt API*, which is dynamically bound to one of the implementations shown in ► Figure 5.28. If more than one implementation is available, then the crypt layer chooses the one that offers the best performance. In particular, on Google Android devices, a system-wide OpenSSL installation might

²⁴Platform-specific byte order in memory

be available, missing, or not supported due to version conflicts. If OpenSSL is not available, then the crypt layer exploits Android's Java API (javax.crypto) by means of callbacks to the platform API layer. All crypt implementations comply with the Environs crypt API specification and make use of platform-specific cryptography APIs to perform cryptographic tasks.

5.4.6.8 Platform-specific API Layer

The platform-specific API layer represents the linchpin for application logic to access framework functionality (cf. ► Figure 5.29). It receives API calls from applications, translates them to commands, manages and delegates them to appropriate framework layers, and informs the caller about the results of a call, whereby results of longer lasting tasks are asynchronously notified to the caller by default, see notification-broker and async-worker in ► Subsection 5.4.6.2.

Responsibilities of platform-specific API layer.

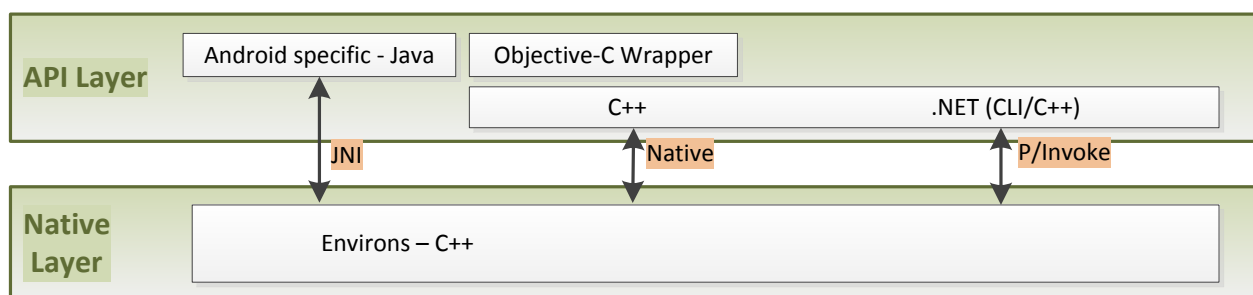


Figure 5.29: The API layer of Environs instances.

As sketched in ► Figure 5.29, only two different code branches are required for the platform-specific layer, that is, a Java branch and a C++ branch. The Java branch targets Google Android devices, whereas the C++ branch targets any other platform. Normally, more dedicated code branches would be required to include platforms such as Microsoft Windows .NET (C#) or Apple iOS / OSX (Objective-C / Swift). However, the design of Environs as shown in ► Figure 5.12 enables supporting both aforementioned (and in theory many other) platforms by means of the C++ code branch. Both platforms may either employ the pure object-oriented C++ API or a platform-specific API as briefly explained in the following.

Platform-specific API layer is composed of two code branches.

For supporting Microsoft Windows .NET platforms, the C++ code branch is compiled as CLI/C++ (that is the C++ language of .NET) to provide framework assemblies, which can be seamlessly integrated into any .NET-based application.

.NET supported by CLI/C++.

Objective-C wrapper
for iOS and OSX.

For supporting Apple iOS / OSX platforms, the C++ code branch is compiled as regular C++ object files enriched with an Objective-C language wrapper. The latter primarily functions as a proxy for C++ objects and connects the internal memory and lifecycle management of Environs with the automatic reference counting mechanism (ARC²⁵) employed by the Objective-C / Swift programming language. The ARC technology enables developers to skip caring for lifecycle management of objects but leave it to the programming language tools. By these ways, the Environs framework provides native Objective-C objects, which also can be easily integrated into applications developed with the more recent and modern programming language Swift²⁶.

5.4.7 Programming APIs

Two kinds of API.

The Environs framework provides two kinds of API to access framework functionality, which is available on every platform, that is (1) a raw API consisting of static C functions with function pointer callbacks and (2) an object API following object-oriented design principles and patterns such as MVC²⁷ or observer pattern (Gamma et al. [1995]). Which one to use depends on the requirements of the designated application and preferences of application developers.

The Raw API. Fast
and resource friendly.

The raw API offers the best performance in terms of speed, latency, resource usage, and communication of results. If performance or system resources are of utmost importance then the raw API suits best, for example, very small devices such as micro-controllers in smart-TVs or IoT-devices. Such devices are usually quite limited in terms of available resources. Hence, the raw API should be the first choice to make such devices part of an application environment (e.g., for home automation [Jin et al., 2014]) and accessible / controllable through other devices in the environment. However, the raw API requires applications to keep track of the states of underlying state machines because only state transitions are communicated through the raw API. In contrast, the object API is based on objects of object-oriented languages to model and keep the states of underlying state machines in order to ease programming. Objects automatically interact with dependent or corresponding objects to mutually keep their states consistent. Thus, developers do not need to know details about internal state machines of the framework but solely rely on behavior exposed by objects. However, the drawback of the object API is slightly reduced performance and higher resource usage due to overhead induced by object-oriented languages.

Complexity of raw API.

The Object API. Easy
to use.

Costs of object API.

²⁵ Automatic Reference Counting

²⁶ <https://swift.org>

²⁷ Model View Controller

5.4.7.1 Raw API

The raw API is basically implemented as a C interface with static functions of the framework libraries. Therefore, any application written in pure C language can directly access the raw API. High-level languages, such as C# or Java realize access to the raw API by means of one static class called *Environs* with static methods.

Implementation of the raw API.

Communication through the raw API happens through callback functions provided by applications. Each function call of the raw API as well as callback function carries numerical parameters to identify a certain state machine within the native layer. For example, each call has at least a number to identify a particular instance of Environs since multiple instances can coexist. In order to connect to a particular device, the API call must provide at least an Environs instance ID and the numerical identifier of the target device. Numerical identifiers of state machines are communicated to applications through callbacks. For example, if a new device appeared in the application environment, then the application is notified about this event together with a numerical identifier that has been assigned to the device's state machine. This approach enables applications to keep track of state machines (e.g., devices, messages, files, sensors, portals, touches, etc.), which are of interest and ignore all other state machines in order to keep resource footprint as low as possible. The drawback, however, is that developers need to know and understand the internals of the Environs framework, for example, which state machines are available or how do state machines depend on other state machine's states.

Using the raw API.

State machines and its identifiers.

5.4.7.2 Object API

The object API is built on top of the raw API and models state machines and their dependencies as well as states and state transitions by means of the object-oriented programming paradigm. Objects have a defined lifecycle and a defined state at any time, which are automatically adapted based on method calls of application code or by evaluation of notifications from native layers.

Under the hood, all objects and its method calls make use of the raw API. However, the objects internally keep track of numerical identifiers and automatically transition their states. By this means, the object API unburdens developers from learning the internals of the framework but just interact with objects. For example, in order to connect to a device, developers just have to call the *Connect* method of a device object. The device object knows its numerical state machine identifier as well as its related Environs instance. The object also knows its current state and performs the connection request only if it's not already connected

Object API makes use of raw API

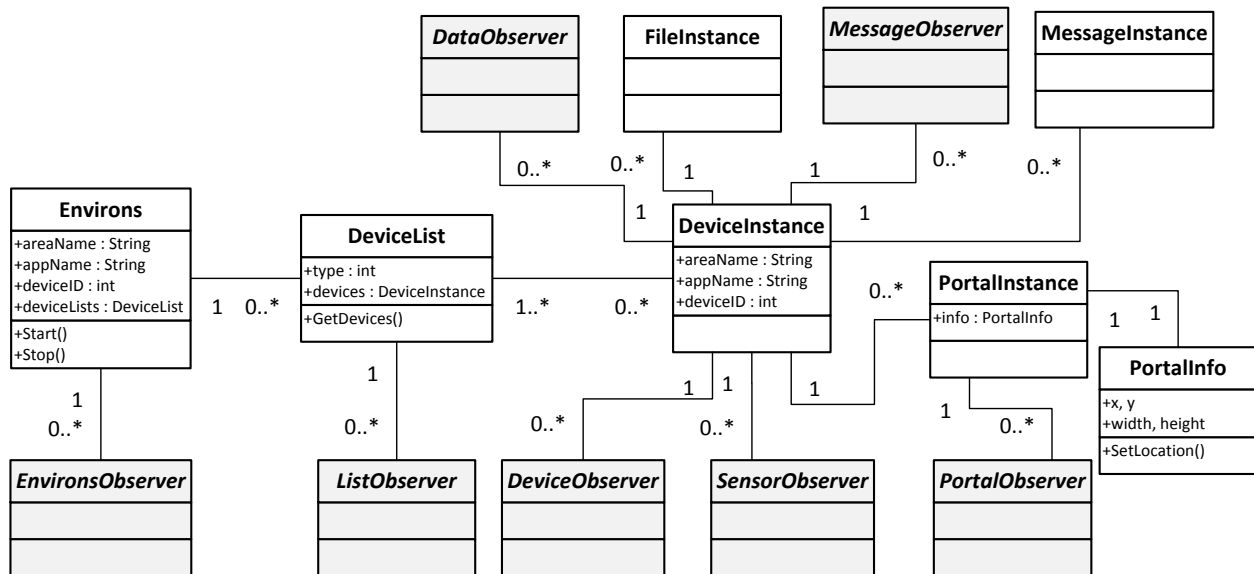


Figure 5.30: Class diagram for the most important objects provided by the object API.

or already within an ongoing connect-process (e.g., requested a previous call or concurrently by the other corresponding device). This simple example alone with the few exceptions shows that the object API is easier to use (in particular for beginners with only few programming experiences) than the raw API because the objects take over all state machine related tasks and constraints, which would otherwise have to be handled by developers.

Asynchronous events
using the observer
pattern.

OBSERVERS. In order to get notified about events and results of asynchronous method calls, application code can attach observer objects to each object (Gamma et al. [1995]). This enables applications to selectively choose notification sources (the objects). In contrast, the raw API requires applications to evaluate all notifications of the native layer and filter out unrelated notifications. Each observer is called back with a reference to the related object as well as associated objects. For example, if a connect-process was successful then the according observer is called back with a reference to the connected object and a notification that indicates the successful connection. If a message was received from a device, then the observer that was attached to the according device object is called back with a reference to the device object together with a message object created for the received message.

What classes of the
object API are
available?

ENVIRONS CLASSES AND OBJECTS. ► Figure 5.30 shows a class diagram with the most important classes used by the object API. Classes that are shown grayed represent observer objects. Such observer objects have to be implemented by developers and can be attached to according API objects. ► Figure 5.30 shows seven important API classes and their association with each other.

ENVIRONS. Each instance of the Environs class manages the lifecycle of the application in a particular application environment. Hence, multiple instances of the Environs class can be created and alive in order to interact with multiple application environments at the same time. The Environs object serves to control the lifecycle of all other related objects. For example, if application code calls the *Stop* method of an Environs object, then all activities of related DeviceInstance objects are also stopped. Under the hood, the DeviceInstance objects' state machine's states are transitioned to the final state, which induces state transitions of all dependent objects. If an interactive portal (represented by a PortalInstance object) is actively streaming, then the PortalInstance object's state is transitioned to the final state, which results in stopping the portal. If a file transfer (represented by a FileInstance object) is ongoing, then the state transition aborts the file transfer. EnvironsObserver objects can be attached to Environs objects in order to get notified if state transitions were successful or have failed.

An Environs object manages lifecycles of related objects.

An Environs object hosts all other objects of a particular Environs instance.

EnvironsObserver objects notify about results of state transitions.

DEVICELIST. Each Environs object is able to manage multiple DeviceList objects and each DeviceList object might manage devices of the same type or of different types. As discussed in ► Subsection 5.4.6.3, devices can be in the same network or on a different network. Hence, DeviceList objects may include devices that are in the same network, in different networks, or in either network. The DeviceList object is automatically adapted by the platform layer based on native layer notifications in order to reflect the current state of available devices at any time. ListObserver objects can be attached to DeviceList objects in order to get notified if new devices have appeared or if one or more of the available devices have vanished.

DeviceList objects show devices in application environments.

ListObserver objects notify about list changes.

DEVICEINSTANCE. Each device in a DeviceList object is represented by a DeviceInstance object, which is the most important object for applications. Interaction with a certain physical device and MSE application within the application environment happens completely through the corresponding DeviceInstance object in the first place, for example sending a message, connecting to the device, or establishing an interactive portal. Because of this rich functionality, a DeviceInstance object supports multiple types of observers dedicated to different types of tasks. The main observer class is the DeviceObserver, which informs application code about all state changes of a DeviceInstance object. For example, to name but a few, whether device connection state changed as well as connection progress in percentage, whether a device has gone or changed its name, whether an interactive portal has been requested/provided as well as the corresponding progress of portal creation steps, or whether the reachability of a device has changed (e.g., disappeared from local network and henceforth available only through a mediator service). All other observer classes will be detailed in the following with their associated platform layer object.

DeviceInstance objects represent devices in application environments.

DeviceInstance objects support multiple observer classes.

MessageInstance objects represent one particular message sent or received by Environs.

DeviceInstance objects keep histories of MessageInstance objects.

MessageObserver objects notify about message events.

FileInstance objects represent one particular data/buffer sent or received by Environs.

DataObservers objects notify about file/buffer events.

PortalInstance objects represent one particular interactive portal.

Accept or deny an interactive portal through PortalInstance objects.

PortalObserver objects notify about portal events.

PortalInfo objects represent the current visual state of interactive portals.

MESSAGEINSTANCE. Each message that was received or sent by a device is represented through a MessageInstance object, which includes the message itself including details about the message such as, the time and date of arrival/creation, the route (mediator server or device end-to-end connection) or whether the message was received or sent. DeviceInstance objects can be instructed to keep a history with all messages, which can be queried by application code, for example, to analyze and build a history of the communication in a chat application. Furthermore, if application code has attached a MessageObserver object to a DeviceInstance object, then the MessageObserver is informed about all message related activities of the particular device by means of MessageInstance objects.

FILEINSTANCE. Data and buffer transfers between devices are significantly different from message exchange as they are not limited to text content. Hence, all data and buffer related activities, as well as progress on related tasks, are represented through FileInstance objects. In order to get notified about such transfers from a particular device, applications have to attach a DataObserver object to the according DeviceInstance object. A new FileInstance object is then created for every incoming or outgoing transfer and provided to application code through DataObserver objects. DataObservers are also notified, for example, if the transfer progress has changed. The FileInstance object carries a percentage value that indicates the amount of data that has already been sent or received. If the transfer has been completed, then the FileInstance object enables direct byte-wise access to the related content.

PORTALINSTANCE. Interactive portals are completely configured and handled through PortalInstance objects, which can be either created on demand by applications or as a result of portal requests by other devices. Each PortalInstance object on one device has a PortalInstance object counterpart on the corresponding interactive portal device. Both corresponding PortalInstance objects (i.e., their underlying state machines in the native layer) interacts with each other in order to configure and establish an interactive portal. For (incoming) portal requests, applications can selectively accept or deny the interactive portal requested from the DeviceInstance object that is associated with each PortalInstance object. If applications created or accepted to a PortalInstance object, PortalObserver objects have to be attached to the PortalInstance objects in order to get notified about state changes of an interactive Portal. PortalInstance objects enable easy and unique configuration and control of interactive portals across all platforms through simple interaction with objects.

PORTALINFO. Each PortalInstance object is associated with exactly one PortalInfo object, which serves two means. First, it always reflects the current position and size of the interactive portal, that is, they are automatically updated and synchronized by the native layer. Second, PortalInfo objects enable applications

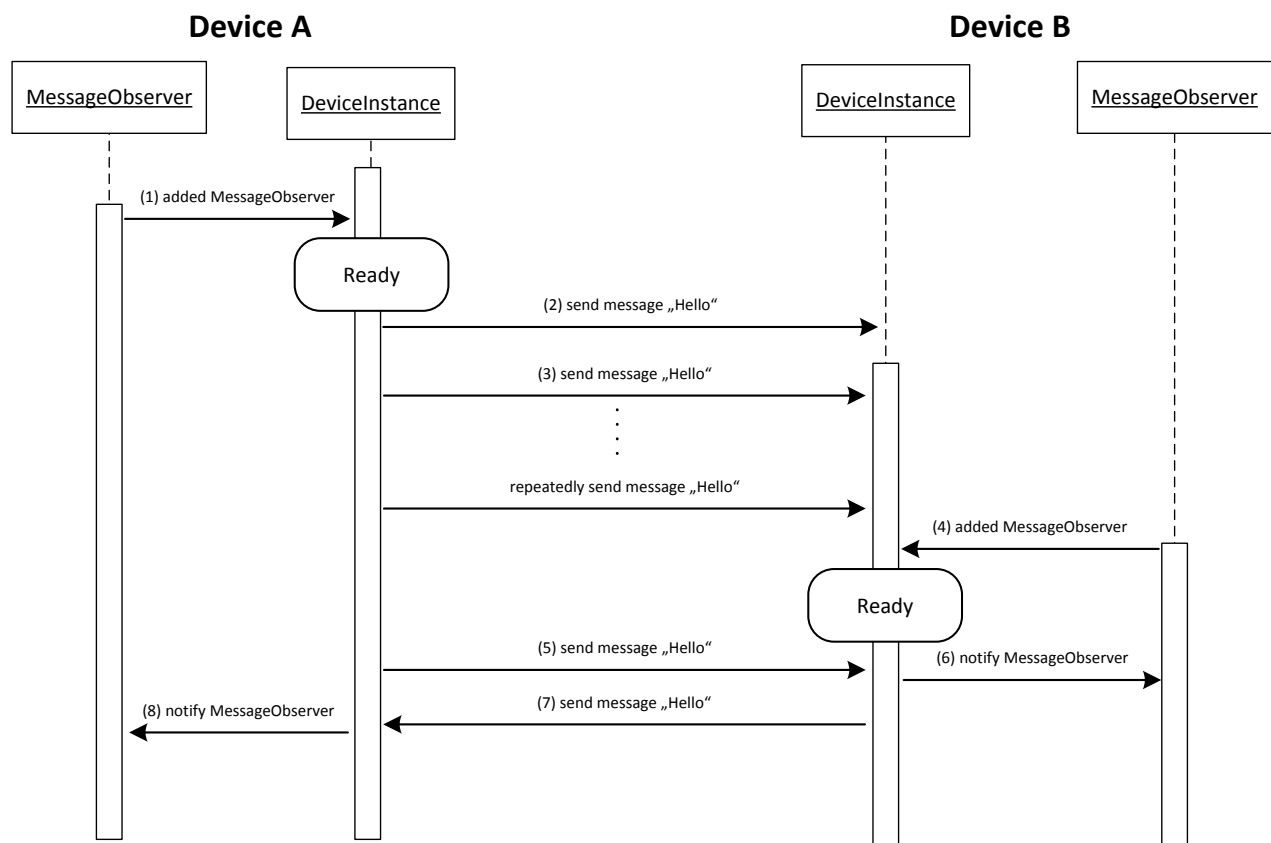


Figure 5.31: Example cases for platform object states with different lifetimes.

to easily control the position and size of an interactive portal if the portal type enables such manipulations.

SYNCHRONOUS SEND/RECEIVE. Applications that realize network communication often make use of synchronous control flow for sending and receiving through so-called network sockets. In order to support such kind of network programming, DeviceInstance objects also provide synchronous send/receive calls in addition to the asynchronous observer approach. By this means, the platform layer enables easy substitution of already available network logic in applications with send/receive calls to DeviceInstance objects, which behave similar to traditional network socket calls.

Support for legacy synchronous control flows for send/receive.

ENVIRONS OBJECT STATE SYNC. During the development of Environs-based MSE applications within student courses, we found that every application had to synchronize the state of platform layer objects with each other, which introduced considerable overhead due to waiting entities that are potentially prone to race condition errors and often resulted in unexpected application behavior. ► Figure 5.31 shows an example for situations that happened quite often. The figure shows two networked devices within an application environment and their

Reduce race condition errors and polling by automatic mutual state machine sync.

according `DeviceInstance` and `MessageObserver` objects. Oriented arrows between the objects indicate their activities as well as message exchange. The reason for the requirement of polling cycles and synchronization are different lifetimes of platform layer objects. Depending on, for example, particular device platforms and device configurations in terms of resource capacity and operating system, platform layer objects start to live at different times. In ► Figure 5.31, the `DeviceInstance` object at device A is created earlier than on device B.

Example for the need of polling entities.

The application on device A first attaches a `MessageObserver` to the `DeviceInstance` object (1) in order to receive messages from device B. Afterwards, device A sends a "hello" message to device B and waits until it responds. However, even though the `Environs` layers on device B handle the hello message from device A, it may happen that the platform layer on device B is still about to create the platform layer object (2). Hence, the message cannot be dispatched to an existing platform layer object and the application on device B has no notice about the hello message. Therefore, device A has to implement a timeout and retry mechanism in order to detect whether device B is ready for communication.

`MessageObserver` objects may or may not be created and attached in time.

Even if the `DeviceInstance` object on device B already exists, the application may not had the time to attach a `MessageObserver` to the `DeviceInstance` object (3). Though the `DeviceInstance` object on device B consumes the received message, there is no way to dispatch the message to the application. As soon as the application on device B has attached a `MessageObserver` object to the `DeviceInstance` object, device B is ready for communication (4). After both `DeviceInstance` objects are in the ready-state, communication can be successfully performed (5, ..., 8).

State synchronization through forward- and backward-propagation in native layer.

The most obvious solution to this problem is to implement polling cycles within application network code with timeouts and retries in order to detect whether communication with a particular device is safe or not. This way of synchronization is inherently inefficient due to the polling activities that run on every device and for every `DeviceInstance` object. However, platform layer objects know their state at any time, that is, whether a communication is possible or not or whether application code has attached an appropriate observer or not. Instead of leaving state synchronization to applications, the platform layer implements an efficient object state synchronization approach based on forward- and backward-propagation. This approach makes use of the asynchronous infrastructure of the `Environs` framework as exemplarily sketched in ► Figure 5.32.

Forward-propagation caused by platform layer objects.

As soon as application code attaches an observer to a `DeviceInstance` object (1, 3), the platform layer forward-propagates (2, 4) the new state to the target device by delegation to the native layer of `Environs`. In turn, the network layer determines the fastest network route to the target device and communicates the

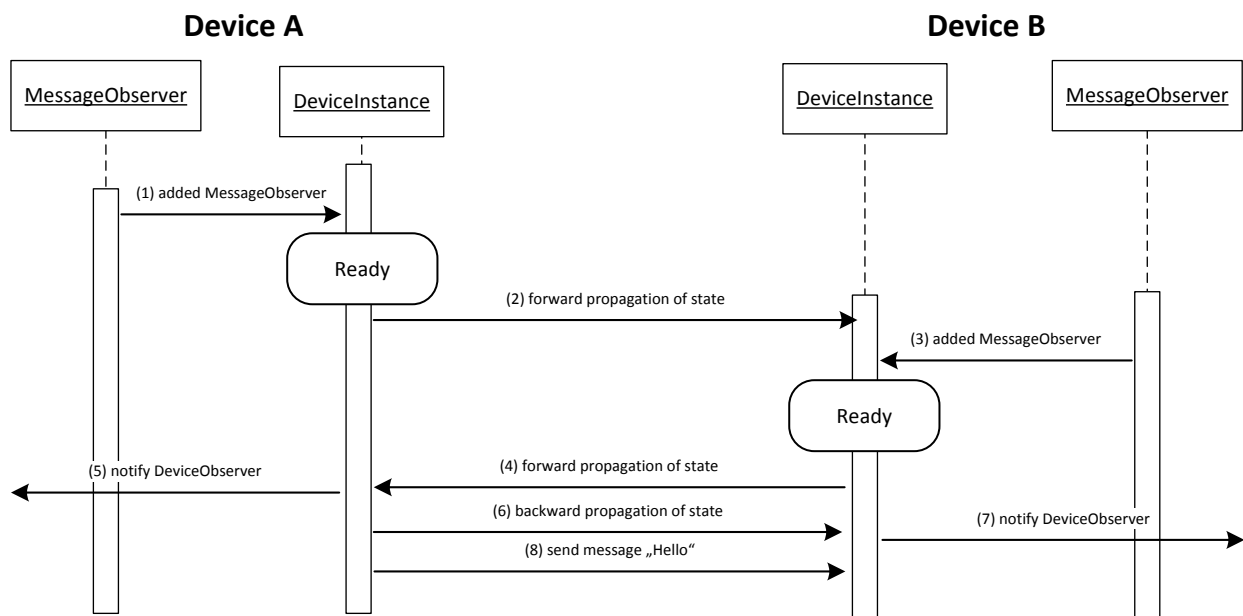


Figure 5.32: Forward- and backward-propagation of object states by the platform layer.

state with the network layer on the target device. If devices are not connected to a mediator server then Environs includes state propagation into the common broadcast communication of the network layer and mediator layer.

Optimal routes determined by network and mediator layer.

As soon as the platform layers update the object state of a `DeviceInstance` object, application code is notified through the according `DeviceObserver` object (5, 7) if one is attached. If the platform layer detects that state updates have been missed or are inconsistent, for example, due to different lifetimes of objects, the platform layer back-propagates object states to corresponding `DeviceInstance` objects (6) in order to keep states in sync.

Backward-propagation caused by consistency checks.

By this means, application code can always query whether `DeviceInstance` objects are ready for communication – that is, messages can be safely dispatched – or not. If not, then the `DeviceObserver` objects will be notified as soon as a `DeviceInstance` object is ready for communication. This approach eliminates the requirement for polling cycles from application code and moves synchronization of object states to the much more efficient asynchronous infrastructure of the Environs framework.

5.5 Chapter Summary

This chapter documented the largest contribution of this dissertation and complements the design space with a design subspace for tabletop interaction in multi-surface environments. Such kind of interaction, also called cross-surface interaction, using digital tabletops is trending in research and expected to become more and more important with increasing importance and proliferation of mobile surface devices.

In order to establish the design subspace for this chapter, we conducted an extensive literature review to determine the current state of research and identify criteria for classifying interactions and related aspects. We then spanned and discussed the design subspace in ► Table 5.1 along the interaction criteria input types, tasks, channels, and base technologies, which can be combined to realize interaction in multi-surface environments.

Similar to the previous chapters, we discussed enabling technologies and reasoned that there is a lack of suitable infrastructure technologies. Hence, this chapter contributed an analysis of the challenges and requirements of an appropriate infrastructure software. Based on the results, the chapter presented the design, architecture, and reference implementation of our open-source software framework called *Environs* that fills the identified gap. *Environs* particularly addresses heterogeneous device ecologies with high requirements on low latency together with flexible support for all kinds of sensors available in modern surface devices. In addition to that, the framework enables high demanding distributed applications that utilize real-time video-based interactive portal channels of the design subspace. The open-source reference framework is publicly available and shall foster advances for cross-surface interactions based on interactive portals in multi-surface environments. So far, it has been used for research prototypes by us [Jin et al., 2014, Dang et al., 2015] and by the University of Waikato [Masoodian et al., 2016].

Summary and Conclusions

THIS chapter breaks down and briefly summarizes the research presented in this dissertation. We recap our contributions and show how these advance the field of tabletop interaction. Finally, we propose areas for future work and directions for promising research not covered in this dissertation, which are made possible by the contributions of this dissertation.

6.1 Design Space for Tabletop Interaction beyond Touch

We motivated this dissertation in ► Chapter 1 by stating and discussing four research problems, which we consider as problems of high importance. Touch input represents the most prominent, thus, most often implemented input mechanism by digital tabletops. We reasoned, however, why it is important to take a broader and comprehensive look at the possibilities for tabletop interaction. Digital tabletops offer much more potential to realize interaction techniques than merely using x/y-coordinates of touch contacts or passive tangible artifacts. If designers and developers attempt to exploit the potential of digital tabletops, they encounter the immediate problem that knowledge about interaction possibilities and according evaluations are spread across the research literature. Often, implementation of those require certain enabling technologies (e.g., tangible artifacts and marker recognition; above the surface interaction and tracking approaches). Depending on the technologies, they pose certain limitations (e.g., precision of finger orientations for FTIR, DI, Capacitance; limitations of depth cameras for above the surface interactions). Without sufficient experiences in the design of tabletop interactions, it would be quite laborious to gather design choices and

How to design
tabletop interaction
beyond touch?

to assess the limitations and applicability of different design choices for certain interaction techniques.

Contribution

This dissertation addresses this problem by providing a comprehensible design space for tabletop interaction beyond touch. As the term design space is used differently with different meanings in the existing literature, ► Chapter 2 clarifies the usage of the term for this dissertation. Also, we discussed existing related design spaces and showed their shortcomings for the problem stated in ► Chapter 1. Within the discussion, we reasoned why (in our design space) interaction techniques and enabling technologies were assigned to the three design subspaces: finger input properties of touch contacts, multimodal and tangible interaction, and interactions in multi-surface environments. All the design subspaces contain techniques and technologies that can be realized separately as well as in combination with each other.

Conclusion

The design space, which is composed of the design subspaces, offers an overview of tabletop interactions beyond touch in relation to enabling technologies, which serve at least three usage scenarios:

1. The design space enables designers and developers to design tabletop interactions for applications based on an overview over available choices. At the same time, they have an overview over required technologies that may serve, for example, for cost estimates that are often an important factor in industrial project planning.
2. In case of designing interactions for already available technology, the design space shows available interaction possibilities that are available and meaningful for particular technologies.
3. By means of the design space, users might acquire knowledge about interaction possibilities of digital tabletops and may explore input mechanisms beyond sole touch input.

In conclusion, all these possibilities help enable tabletop interaction beyond sole touch and create an awareness of tabletop interaction that is not limited to touch input through x/y-coordinates.

Problems of high importance.

By analyzing the three design subspaces, we identified neglected and missing research in each design subspace, which we consider as research problems of high importance. Those research problems were introduced in ► Chapter 1 as individual problem statements. The remaining chapters of this dissertation discussed each of the design subspaces followed by our contributions for the according identified problem spaces.

6.2 Finger Input Properties of Touch Contacts

A sole touch contact results in a contact area that offers more interaction related properties than merely an x/y-coordinate. The availability of those properties, however, strongly depends on the underlying sensor technologies. Thus, ► Chapter 3 investigated surface sensor technologies that are used to sense contact areas and analyzed finger input properties that are provided through touch contact areas in relation to sensor technologies. We then spanned the chapter's design subspace (cf. ► Table 3.1) with enabling technologies and finger input properties as the dimensions and showed the availability of each combination within the design subspace. Within the analysis, we also discussed limitations, precision, and reliability of finger input properties for different technologies if those were significantly different from other technologies. For each of the finger input properties, we discussed interaction techniques that have been investigated and evaluated in related works, which provide design choices for interaction design. For the Diffused Illumination technology, we contributed our experiences and detailed descriptions for an extension called Pulsed Diffused Illumination. This extension improves the contrast of sensor data in order to increase reliability and precision of touch contact recognition.

Interaction possibilities through touch contact areas.

Contribution: Design subspace.

In this design subspace, we identified that finger orientation and hand distinction were under-researched / missing finger input properties in the literature. We addressed those finger input properties with original contributions in dedicated sections. ► Section 3.3 investigated the potential of finger orientation for tabletop interaction and presented interaction techniques that utilize the finger orientation. Subsequently, we presented an efficient computer-vision algorithm to determine finger orientations through the Diffused Illumination technology. We further presented an evaluation that proved that our approach achieves high precision of the recognition. Our work on finger orientation completed the availability of this finger input property in the design subspace for all other optical-based enabling technologies beyond FTIR. Designers and developers, therefore, can consider interaction techniques based on finger orientations for all optical-based surface sensor approaches as available. ► Section 3.4 investigated the potential of hand distinction for tabletop interaction and presented interaction techniques that rely on hand distinction. In particular, digital tabletops afford for multi-touch, bi-manual, or collaborative interaction in which touch contact areas may originate from multiple hands or from the same hand. Hand distinction addresses those interactions by providing this information to applications. This information can, for example, be used to solve conflicts when touch contacts on the same object originate from different users. Different effects of gestures may be realized depending on whether only one hand has performed the gesture or two or even more hands have performed the gesture. After the discussion

Contribution: Finger orientation.

Conclusion.

Contribution: Hand distinction.

Conclusion.

of hand distinction, we presented a heuristic approach for hand distinction that utilizes only x/y-coordinates and finger orientations of touch contacts. We further presented an evaluation and according results for the high precision of the recognition approach. Our work on hand distinction completed the availability of this finger input property in the design subspace with an approach that does not require additional instrumentation of digital tabletops. Designers and developers, therefore, can consider interaction techniques based on hand distinction for all presented surface sensor approaches as available. However, we also showed that the precision of recognition depends on the precision of finger orientation detection, which is different for different technologies. The best results can be achieved through Diffused Illumination based digital tabletops.

Conclusion.

Overall, this design subspace – completed with our contributions to finger input properties – enables designers and developers to design tabletop interaction beyond touch without the need for further instrumentation with additional hardware. Interaction techniques that are based on design choices provided by this design subspace only utilize the surface sensor technologies that are required for digital tabletops anyway. In conclusion, the interaction possibilities showed in this design subspace alone enable designers and developers to realize tabletop interaction beyond touch.

6.3 Multimodal and Tangible Interaction

Interaction techniques that physically go beyond the tabletop surface.

Design subspace: Multimodal and tangible interaction.

Contribution: Design subspace.

The second design subspace modeled interaction possibilities for digital tabletops that are spatially located beyond the tabletop surface. In contrast, the previous design subspace consolidated interaction techniques that are spatially located on the tabletop surface. If we go beyond the tabletop surface, tabletop interaction is characterized by additional input devices, which are represented in the literature as multimodal tabletop and tabletop tangible interaction. In principle, tabletop tangible interaction can be considered as part of multimodal tabletop interaction. However, the quantity of research for tabletop tangible interaction is comparable or even higher than the quantity of research for all modalities together within multimodal tabletop interaction. Thus, in order to adequately represent tangible tabletop interaction and provide a realistic view of the possibilities, ► Chapter 4 discussed both topics in separate sections with dedicated contributions. Both sections introduced into the research fields by clarifying terms and giving definitions for those, which is particularly important for this chapter as terms, such as modality or mode, were regularly used in an ambiguous and vague manner. We then provided the design subspace for multimodal tabletop interaction and tabletop tangible interaction in ► Table 4.1 and provided a novel classification scheme for tabletop tangibles in ► Figure 4.11.

► Table 4.1 listed interaction modalities together with according interaction channels, interaction codes, and enabling technologies, which were then discussed in detail afterwards. The discussions included concrete interaction techniques, limitations of the modalities and technologies, and corresponding works in the literature. The novel aspect of our design subspace is the inclusion and discussion of modalities, such as kinesthetics, proprioception, or physiological signals, in the context of multimodal tabletop interaction, which gives a comprehensible overview over multimodal tabletop interaction.

Afterwards, ► Section 4.2 presented a classification scheme in ► Figure 4.11 to categorize tangible artifacts, which includes novel and recent kinds of active actuated tangibles. We discussed the distinguishing criteria and limitations of implementations together with significant research works that investigated appropriate interaction techniques.

In this design subspace, we identified that embodied interaction in combination with active actuated tangibles has not been considered in the literature. Therefore, we contributed original research that proposed an approach to realize such kind of interaction with detailed descriptions of hardware, crafting of the system as well as required software. We further conducted a user study to evaluate how users cope with such novel interaction mechanisms and collected user feedback regarding their user experience. The results showed that users quickly understood the interaction mechanism and could cope with the system. Overall, users were positively minded and enjoyed to remotely control physical objects by means of body movements mediated through digital tabletops. However, the results also revealed that latency is an important factor for such kind of interaction. The shorter the duration between user input and system reaction through activity of the actuated tangibles, the better the user experience. Our whole system, that is, hardware and software, was already optimized for low latency, but had a limiting component that was used for user tracking. We employed a Microsoft Kinect depth camera for user tracking, which (together with the gesture recognition approach) induced the largest part of the overall latency. The remaining part of the overall latency was negligible small. Nevertheless, our research is regarded as a first step towards such kind of interaction with promising results.

In conclusion, this design subspace showed available design choices for tabletop interactions that go beyond the tabletop surface, that is, for example, above or surrounding a digital tabletop. Designers and developers were given options for interaction design that may be realized individually or in combination with techniques from the other design subspace. As a result, all the options for interaction design lead to tabletop interaction techniques that go beyond touch input.

Contribution:
Embodied interaction
with active actuated
tangibles.

Conclusion.

Conclusion.

6.4 Interaction in Multi-Surface Environments

In ► Chapter 5, we further expanded the interaction space to interaction techniques that are spanned across multiple surfaces. In contrast, the previous design space considered interaction techniques that were spatially distributed above or around a digital tabletop. We first introduced into potential interaction spaces that can be realized with nowadays mobile surface devices, such as smartphones, tablets, or smartwatches. Together with digital tabletops, such a device ecology span so-called tabletop centered multi-surface environments (MSE), which are the subject of research of ► Chapter 5.

This chapter contributed a thorough literature review that provided the current state of research for interaction techniques in tabletop centered MSEs. Based on an analysis of the literature, we presented and discussed a design subspace for interaction techniques in such MSEs. The structure of the design subspace is different from the previous design subspaces because enabling technologies in terms of hardware as well as interaction techniques through individual devices were basically the same as were introduced in the previous design subspaces. Hence, we structured the design subspace in ► Table 5.1 according to criteria that adequately distinguish interaction techniques in MSEs, in particular those that are spanned across multiple surfaces.

Contribution:
Literature review and
state of research for
interactions in MSEs.

Design subspace:
Interaction in MSEs.

In this design subspace, we identified the lack of suitable concepts and architectures for infrastructure frameworks to enable interaction techniques and applications for nowadays heterogeneous surface device ecologies. Hence, the remainder of the chapter addressed this shortcoming and contributed the design, concepts, and the architecture of our MSE infrastructure framework "Environs". We first analyzed the challenges and requirements of MSE frameworks in consideration of heterogeneous device ecologies, means for interaction provided by mobile surface devices, and factors that influence user experience. Afterwards, we presented our MSE framework in detail and discussed how the design, the concepts, and the architecture address the identified challenges and requirements. In addition to that, our MSE framework implements so-called interactive portals, which enable demanding real-time video-based interactive applications.

Contribution:
Multi-surface
infrastructure
framework Environs.

In conclusion, this design subspace showed available design choices for tabletop interactions in MSEs that cross the boundary of one surface and involves multiple surface based devices. Designers and developers were given options for interaction design that may be realized individually or in combination with techniques from both previous design subspaces. As a result, our design space comprised of its design subspaces provide options for interaction design that spatially begin on a touch sensor surface, extend to above and/or around a digital tabletop, and involve multiple surface based devices. Through these

Conclusion.

design options, designers and developers have a comprehensive overview over tabletop interaction and are able to enable interaction designs that lead to tabletop interaction techniques that go beyond touch input.

6.5 Future Work

This dissertation presented a design space for tabletop interaction beyond touch that, in the first place, serves to assist designers and developers. At the same time, the design space enables researchers to identify and spot directions for further research endeavors. For example, research may address approaches to extend finger input properties of ► Table 3.1 or novel kinds of interactions beyond those multi-surface interactions identified in ► Table 5.1.

In this section, however, we briefly report on two further research directions supported by our experiences with the dedicated research works that we contributed to each design subspace.

During the review of multimodal tabletop interactions, we found that most research works combine touch input with one additional modality (e.g., speech, pen, pointing, eye gaze). The design space revealed the need for multimodal theories that systematically analyze which combination of modalities would make sense and how modalities can be combined in a meaningful manner. Such further research efforts may benefit from results of multimodal research in which no tabletop surfaces were included or were not a central part in the interaction (e.g., [Wasinger, 2006]). There is already research in this direction, however, considering only parts of the design subspace for multimodal tabletop interaction. For example, the dissertation of Wasinger [2006] investigated combinations of modalities for multimodal interaction with mobile devices. Another related dissertation from Tse [2007] analyzed combinations of speech commands, gestures, and touch input. Our design subspaces for multimodal tabletop interaction and multi-surface environments could serve as a guide for a further dissertation that analyzes and investigates combinations of interaction techniques in multi-surface environments.

In the review of multi-surface environment literature, we found that the locations of surfaces and users were often an important aspect for realizing novel interaction techniques. A systematic analysis of approaches and technological means to determine and communicate locations within multi-surface environments would provide a fruitful basis for further concepts that can be used in MSE frameworks. With such location-aware frameworks, interaction approaches of the multimodal tabletop interaction/finger input properties design subspace that employ location information (e.g., from finger orientation or speech commands)

would enrich cross-surface interactions in multi-surface environments and provide many directions for further studies. For example, the (finger) orientation-based selection techniques (as we have presented in ► Section 3.3) could be extended to remote surfaces in MSEs for selection or view switching similar to gaze-based techniques [Pfeuffer et al., 2015, Voelker et al., 2015b]. Techniques based on speech commands, such as proposed in Tse et al. [2006], could be extended to MSEs. If an MSE application would know the locations and perspectives of users and devices, such information could be used to disambiguate between different target surfaces.

“Iucundi Acti Labores”

— Cicero

PUBLICATIONS OF THE AUTHOR

Publications with content from this dissertation

Long paper

Dang, Chi Tai, Straub, M., and André, E. (2009). Hand distinction for multi-touch tabletop interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 101-108, New York, NY, USA. ACM.

<http://dx.doi.org/10.1145/1731903.1731925>

Dang, Chi Tai and André, E. (2011). Usage and recognition of finger orientation for multi-touch tabletop interaction. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part III*, INTERACT 2011, pages 409-426, Berlin, Heidelberg. Springer-Verlag.

http://dx.doi.org/10.1007/978-3-642-23765-2_28

Dang, Chi Tai and André, E. (2013). TabletopCars: interaction with active tangible remote controlled cars. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, TEI '13, pages 33-40, New York, NY, USA. ACM.

<http://dx.doi.org/10.1145/2460625.2460630>

Dang, Chi Tai and André, E. (2014). A framework for the development of multi-display environment applications supporting interactive real-time portals. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, pages 45-54, New York, NY, USA. ACM.

<http://dx.doi.org/10.1145/2607023.2607038>

Publications related to work in this dissertation

Workshops

Dang, Chi Tai, Masoodian, M. and André, E. (2015). Private Focus Portals to Shared Energy Visualizations. Adjunct Conference in Proceedings of *INTERACT 2015, INTERACT Workshop on Fostering Smart Energy Applications*, FSEA 2015 (Bamberg, Germany, 15 September), University of Bamberg Press.

http://dx.doi.org/10.1007/978-3-319-22723-8_88

Poster

Dang, Chi Tai and André, E. (2010). Surface-poker: multimodality in tabletop games. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 251-252, New York, NY, USA. ACM.

<http://dx.doi.org/10.1145/1936652.1936701>

Jin, Y., **Dang, Chi Tai**, Prehofer, C., and André, E. (2014). A multi-display system for deploying and controlling home automation. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 399-402, New York, NY, USA. ACM.

<http://dx.doi.org/10.1145/2669485.2669553>

Klein, A., Ljaci, N., Budde, A., **Dang, Chi Tai**, and André, E. (2008). Interaction via tangibles and virtuals in a tabletop application. Poster at the IEEE Tabletop 2008 Workshop (Conference DVD), Amsterdam, Netherlands.

Other publications

Journal

Kistler, F., Endrass, B., Damian, I., **Dang, Chi Tai**, and André, E. (2012). Natural interaction with culturally adaptive virtual characters. *Journal on Multimodal User Interfaces*, pages 1-9.

<http://dx.doi.org/10.1007/s12193-011-0087-z>

Long paper

Schmidt, H., **Dang, Chi Tai**, Gessler, S., and Hauck, F. J. (2009). Model-driven development of adaptive applications with self-adaptive mobile processes. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*, OTM '09, pages 726-743, Berlin, Heidelberg. Springer-Verlag.

http://dx.doi.org/10.1007/978-3-642-05148-7_51

Schmidt, H., **Dang, Chi Tai**, and Hauck, F. (2007). Proxy-based security for the session initiation protocol (SIP). In *Second International Conference on Systems and Networks Communications*, 2007. ICSNC 2007., pages 42-42.

<http://dx.doi.org/10.1109/ICSNC.2007.64>

BIBLIOGRAPHY

- Abad, Z. S. H., Anslow, C., and Maurer, F. (2014). Multi surface interactions with geospatial data: A systematic review. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 69–78, New York, NY, USA. ACM.
- Adachi, T., Koura, S., Shibata, F., and Kimura, A. (2013). Forearm menu: Using forearm as menu widget on tabletop system. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 333–336, New York, NY, USA. ACM.
- Agrawala, M., Beers, A. C., McDowall, I., Fröhlich, B., Bolas, M., and Hanrahan, P. (1997). The two-user responsive workbench: Support for collaboration through individual views of a shared space. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 327–332, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- Ajaj, R., Jacquemin, C., and Vernier, F. (2009). Rvdt: a design space for multiple input devices, multiple views and multiple display surfaces combination. In *Proceedings of the 2009 international conference on Multimodal interfaces*, ICMI-MLMI '09, pages 269–276, New York, NY, USA. ACM.
- Al-Megren, S., Kharrufa, A., Hook, J., Holden, A., Sutton, S., and Olivier, P. (2015). *Comparing Fatigue When Using Large Horizontal and Vertical Multi-touch Interaction Displays*, pages 156–164. Springer International Publishing, Cham.
- Anderson, G., Doherty, R., and Ganapathy, S. (2011). User perception of touch screen latency. In Marcus, A., editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, volume 6769 of *Lecture Notes in Computer Science*, pages 195–202. Springer Berlin Heidelberg.
- André, E. and Martin, J.-C. (2014). Multimodal systems. *Mitkov, R., ed.: The Oxford Handbook of Computational Linguistics 2nd edition*, Oxford University Press, pages 1–15.
- André, E., Martin, J.-C., Lingenfelser, F., and Wagner, J. (2013). Multimodal fusion in human-agent dialogue. *Coverbal Synchrony in Human-Machine Interaction*, 10:387–410.
- Ardito, C., Buono, P., Costabile, M. F., and Desolda, G. (2015). Interaction with large displays: A survey. *ACM Comput. Surv.*, 47(3):46:1–46:38.
- Arif, A. S. and Stuerzlinger, W. (2013). Pseudo-pressure detection and its use in predictive text entry on touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 383–392, New York, NY, USA. ACM.
- Bachl, S., Tomitsch, M., Kappel, K., and Grechenig, T. (2011). The effects of personal displays and transfer techniques on collaboration strategies in multi-touch based multi-display environments. In *Proceedings of the International conference on Human-Computer Interaction – INTERACT 2011*, volume 6948 of *Lecture Notes in Computer Science*, pages 373–390. Springer Berlin Heidelberg.
- Bachour, K., Kaplan, F., and Dillenbourg, P. (2010). An interactive table for supporting participation balance in face-to-face collaborative learning. *IEEE Transactions on Learning Technologies*, 3(3):203–213.
- Bakker, S., Vorstenbosch, D., van den Hoven, E., Hollemans, G., and Bergman, T. (2007). Weathergods: tangible interaction in a digital tabletop game. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 151–152, New York, NY, USA. ACM.
- Balakrishnan, R. and Hinckley, K. (1999). The role of kinesthetic reference frames in two-handed input performance. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, pages 171–178, New York, NY, USA. ACM.
- Banerjee, A., Burstyn, J., Girouard, A., and Vertegaal, R. (2011). Pointable: An in-air pointing technique to manipulate out-of-reach targets on tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 11–20, New York, NY, USA. ACM.

- Barnes, G. (1993). A method for implementing lock-free shared-data structures. In *Proceedings of the Fifth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '93, pages 261–270, New York, NY, USA. ACM.
- Barsalou, L. W. (2008). Grounded cognition. *Annual Review of Psychology*, (59):617–645.
- Baudisch, P., Good, N., and Stewart, P. (2001). Focus plus context screens: combining display technology with visualization techniques. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 31–40, New York, NY, USA. ACM.
- Beaudouin-Lafon, M. and Mackay, W. (2003). The human-computer interaction handbook. chapter Prototyping Tools and Techniques, pages 1006–1031. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.
- Begel, A., Garcia, D. D., and Wolfman, S. A. (2004). Kinesthetic learning in the classroom. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 183–184, New York, NY, USA. ACM.
- Bellucci, A., Malizia, A., and Aedo, I. (2014). Light on horizontal interactive surfaces: Input space for tabletop computing. *ACM Comput. Surv.*, 46(3):32:1–32:42.
- Benko, H., Saponas, T. S., Morris, D., and Tan, D. (2009). Enhancing input on and above the interactive surface with muscle sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 93–100, New York, NY, USA. ACM.
- Benko, H., Wilson, A. D., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1263–1272, New York, NY, USA. ACM.
- Bérard, F. and Blanch, R. (2013). Two touch system latency estimators: High accuracy and low overhead. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 241–250, New York, NY, USA. ACM.
- Bernsen, N. O. (1993). Modality theory in support of multimodal interface design. In *ERCIM Workshop on Multimodal Human-Computer Interaction*, pages 37–44.
- Bezerianos, A. and Balakrishnan, R. (2005). View and space management on large displays. *IEEE Comput. Graph. Appl.*, 25(4):34–43.
- Bianchi-Berthouze, N., Kim, W. W., and Patel, D. (2007). Does body movement engage you more in digital game play? and why? In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 102–113, Berlin, Heidelberg. Springer.
- Biehl, J. T., Baker, W. T., Bailey, B. P., Tan, D. S., Inkpen, K. M., and Czerwinski, M. (2008). Impromptu: A new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 939–948, New York, NY, USA. ACM.
- Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 73–80, New York, NY, USA. ACM.
- Bischof, M., Conradi, B., Lachenmaier, P., Linde, K., Meier, M., Pötzl, P., and André, E. (2008). Xenakis: Combining tangible interaction with probability-based musical composition. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction*, TEI '08, pages 121–124, New York, NY, USA. ACM.
- Biskjaer, M. M., Dalsgaard, P., and Halskov, K. (2014). A constraint-based understanding of design spaces. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, pages 453–462, New York, NY, USA. ACM.
- Blattner, M. and Glinert, E. (1996). Multimodal integration. *MultiMedia, IEEE*, 3(4):14–24.
- Bolanowski, Jr., S. J., Gescheider, G. A., Verrillo, R. T., and Checkosky, C. M. (1988). Four channels mediate the mechanical aspects of touch. *Acoustical Society of America Journal*, 84:1680–1694.

- Bolt, R. A. (1980). Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14(3):262–270.
- Bonnet, D., Appert, C., and Beaudouin-Lafon, M. (2013). Extending the vocabulary of touch events with thumbrock. In *Proceedings of Graphics Interface 2013*, GI '13, pages 221–228, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.
- Boring, S., Baur, D., Butz, A., Gustafson, S., and Baudisch, P. (2010). Touch projector: mobile interaction through video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296, New York, NY, USA. ACM.
- Boring, S., Ledo, D., Chen, X. A., Marquardt, N., Tang, A., and Greenberg, S. (2012). The fat thumb: Using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 39–48, New York, NY, USA. ACM.
- Born, R. T. and Bradley, D. C. (2005). Structure and function of visual area mt. *Annual Review of Neuroscience*, 28(1):157–189. PMID: 16022593.
- Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. (2008). Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pages 154–161, New York, NY, USA. ACM.
- Brandl, P., Leitner, J., Seifried, T., Haller, M., Doray, B., and To, P. (2009). Occlusion-aware menu design for digital tabletops. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 3223–3228, New York, NY, USA. ACM.
- Brave, S., Ishii, H., and Dahley, A. (1998). Tangible interfaces for remote collaboration and communication. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, pages 169–178, New York, NY, USA. ACM.
- Brewster, S. A. and Hughes, M. (2009). Pressure-based text entry for mobile devices. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 9:1–9:4, New York, NY, USA. ACM.
- Bruder, G., Steinicke, F., and Stuerzlinger, W. (2013). *Touching the Void Revisited: Analyses of Touch Behavior on and above Tabletop Surfaces*, pages 278–296. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Buchmann, V., Violich, S., Billingham, M., and Cockburn, A. (2004). Fingartips: Gesture based direct manipulation in augmented reality. In *Proceedings of the 2Nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '04, pages 212–221, New York, NY, USA. ACM.
- Buxton, B. (2007). Multi-touch systems that i have known and loved.
- Calife, D., Bernardes, Jr., J. a. L., and Tori, R. (2009). Robot arena: An augmented reality platform for game development. *Comput. Entertain.*, 7(1):11:1–11:26.
- Cao, X., Wilson, A., Balakrishnan, R., Hinckley, K., and Hudson, S. (2008). Shapetouch: Leveraging contact shape on interactive surfaces. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 129–136.
- Chan, L.-W., Kao, H.-S., Chen, M. Y., Lee, M.-S., Hsu, J., and Hung, Y.-P. (2010). Touching the void: Direct-touch interaction for intangible displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2625–2634, New York, NY, USA. ACM.
- Cheng, K., Li, J., and Müller-Tomfelde, C. (2012). Supporting interaction and collaboration on large displays using tablet devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 774–775, New York, NY, USA. ACM.
- Cheung, V., Watson, D., Vermeulen, J., Hancock, M., and Scott, S. (2014). Overcoming interaction barriers in large public displays using personal devices. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 375–380, New York, NY, USA. ACM.

- Ch'ng, E. and Cooke, N. (2015). *User Study on 3D Multitouch Interaction (3DMi) and Gaze on Surface Computing*, pages 425–433. Springer International Publishing, Cham.
- Chokshi, A., Seyed, T., Marinho Rodrigues, F., and Maurer, F. (2014). eplan multi-surface: A multi-surface environment for emergency response planning exercises. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces, ITS '14*, pages 219–228, New York, NY, USA. ACM.
- Cincuegrani, S. M., Jordà, S., and Våljamäe, A. (2016). Physiopucks: Increasing user motivation by combining tangible and implicit physiological interaction. *ACM Trans. Comput.-Hum. Interact.*, 23(1):4:1–4:22.
- Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. (1997). Quickset: Multimodal interaction for distributed applications. In *Proceedings of the Fifth ACM International Conference on Multimedia, MULTIMEDIA '97*, pages 31–40, New York, NY, USA. ACM.
- Collberg, C., Kobourov, S., Kobes, S., Smith, B., Trush, S., and Yee, G. (2003). Tetratetris: A study of multi-user touch-based interaction using diamondtouch. INTERACT'03.
- Colley, A., Häkkinä, J., Schöning, J., Daiber, F., Steinicke, F., and Krüger, A. (2015). Touch the 3rd dimension! understanding stereoscopic 3d touchscreen interaction. In Wyeld, T., Calder, P., and Shen, H., editors, *Computer-Human Interaction. Cognitive Effects of Spatial Interaction, Learning, and Ability: 25th Australian Computer-Human Interaction Conference, OzCHI 2013, Adelaide, SA, Australia, November 25-29, 2013. Revised and Extended Papers*, pages 47–67, Cham. Springer International Publishing.
- Dang, C. T. and André, E. (2010). Surface-poker: multimodality in tabletop games. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS '10*, pages 251–252, New York, NY, USA. ACM.
- Dang, C. T. and André, E. (2011). Usage and recognition of finger orientation for multi-touch tabletop interaction. In *Proceedings of the 13th IFIP TC 13 International conference on Human-computer Interaction - Volume Part III, INTERACT'11*, pages 409–426, Berlin, Heidelberg. Springer-Verlag.
- Dang, C. T. and André, E. (2013). Tabletopcars: interaction with active tangible remote controlled cars. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, TEI '13*, pages 33–40, New York, NY, USA. ACM.
- Dang, C. T. and André, E. (2014). A framework for the development of multi-display environment applications supporting interactive real-time portals. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '14*, pages 45–54, New York, NY, USA. ACM.
- Dang, C. T., Masoodian, M., and André, E. (2015). Private focus portals to shared energy visualizations, fostering smart energy applications. In *Adjunct Conference Proceedings of INTERACT 2015*, volume 9299 of *Human-Computer Interaction - Lecture Notes in Computer Science*, pages 657–658. Springer International Publishing.
- Dang, C. T., Straub, M., and André, E. (2009). Hand distinction for multi-touch tabletop interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS '09*, pages 101–108, New York, NY, USA. ACM.
- Darken P., R. and Peterson, B. (2014). Spatial orientation, wayfinding, and representation. pages 467–491.
- Dietz, P. and Leigh, D. (2001). Diamondtouch: A multi-user touch technology. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, UIST '01*, pages 219–226, New York, NY, USA. ACM.
- Dippon, A., Wiedermann, N., and Klinker, G. (2012). Seamless integration of mobile devices into interactive surface environments. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces, ITS '12*, pages 331–334, New York, NY, USA. ACM.
- Do-Lenh, S., Kaplan, F., Sharma, A., and Dillenbourg, P. (2009). Multi-finger interactions with papers on augmented tabletops. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, TEI '09*, pages 267–274, New York, NY, USA. ACM.

- Doeweling, S., Tahiri, T., Sowinski, P., Schmidt, B., and Khalilbeigi, M. (2013). Support for collaborative situation analysis and planning in crisis management teams using interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS '13*, pages 273–282, New York, NY, USA. ACM.
- Dohse, K. C., Dohse, T., Still, J. D., and Parkhurst, D. J. (2008). Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *ACHI '08: Proceedings of the First International Conference on Advances in Computer-Human Interaction*, pages 297–302. IEEE Computer Society.
- Domova, V., Vartiainen, E., Azhar, S., and Ralph, M. (2013). An interactive surface solution to support collaborative work onboard ships. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces, ITS '13*, pages 265–272, New York, NY, USA. ACM.
- Domova, V., Vartiainen, E., and Englund, M. (2014). Designing a remote video collaboration system for industrial settings. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces, ITS '14*, pages 229–238, New York, NY, USA. ACM.
- Echtler, F., Huber, M., and Klinker, G. (2008). Shadow tracking on multi-touch tables. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '08*, pages 388–391, New York, NY, USA. ACM.
- Echtler, F., Sielhorst, T., Huber, M., and Klinker, G. (2009). A short guide to modulated light. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, TEI '09*, pages 393–396, New York, NY, USA. ACM.
- EloTouch, S. (2015). Elotouch resistive touch technologies (accessed 2015/08). http://www.elotouch.com/Technologies/compare_resist.asp.
- Esenther, A. and Ryall, K. (2006). Fluid dtmouse: better mouse support for touch-based interactions. In *Proceedings of the working conference on Advanced visual interfaces, AVI '06*, pages 112–115, New York, NY, USA. ACM.
- Fikkert, W., Hakvoort, M., van der Vet, P., and Nijholt, A. (2009). Experiences with interactive multi-touch tables. In Nijholt, A., Reidsma, D., and Hondorp, H., editors, *Intelligent Technologies for Interactive Entertainment*, volume 9 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 193–200. Springer Berlin Heidelberg.
- Fitzmaurice, G. W. (1996). *Graspable User Interfaces*. PhD thesis, University of Toronto.
- Fitzmaurice, G. W., Ishii, H., and Buxton, W. A. S. (1995). Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95*, pages 442–449, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- Fogtmann, M. H., Fritsch, J., and Kortbek, K. J. (2008). Kinesthetic interaction: Revealing the bodily potential in interaction design. In *Proceedings of the 20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat, OZCHI '08*, pages 89–96, New York, NY, USA. ACM.
- Ford, B., Srisuresh, P., and Kegel, D. (2005). Peer-to-peer communication across network address translators. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05*, pages 13–13, Berkeley, CA, USA. USENIX Association.
- Forlines, C. and Lilien, R. (2008). Adapting a single-user, single-display molecular visualization application for use in a multi-user, multi-display environment. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '08*, pages 367–371, New York, NY, USA. ACM.
- Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07*, pages 647–656, New York, NY, USA. ACM.
- Frei, P., Su, V., Mikhak, B., and Ishii, H. (2000). Curlybot: Designing a new class of computational toys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '00*, pages 129–136, New York, NY, USA. ACM.

- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Gescheider, G. A., Bolanowski, S. J., Pope, J. V., and Verrillo, R. T. (2002). A four-channel analysis of the tactile sensitivity of the fingertip: frequency selectivity, spatial summation, and temporal summation. *Somatosensory & Motor Research*, 19(2):114–124. PMID: 12088385.
- Gibson, B. M. and Mair, R. (2016). A pathway for spatial memory encoding. *Learning & Behavior*, 44(2):97–98.
- Gibson, J. (1977). The theory of affordances. In Shaw, R. and Bransford, J., editors, *Perceiving, Acting, and Knowing: Toward and Ecological Psychology*, pages 62–82. Erlbaum, Hillsdale, NJ.
- Gjerlufsen, T., Klokmoose, C. N., Eagan, J., Pillias, C., and Beaudouin-Lafon, M. (2011). Shared substance: Developing flexible multi-surface applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3383–3392, New York, NY, USA. ACM.
- Goldinger, S. D., Papesh, M. H., Barnhart, A. S., Hansen, W. A., and Hout, M. C. (2016). The poverty of embodied cognition. *Psychonomic Bulletin & Review*, pages 1–20.
- Greenhalgh, T., Peacock, R., et al. (2005). Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *Bmj*, 331(7524):1064–1065.
- Grossman, T. and Wigdor, D. (2007). Going deeper: a taxonomy of 3d on the tabletop. In *Tabletop*, pages 137–144. IEEE Computer Society.
- Grote, C., Segreto, E., Okerlund, J., Kincaid, R., and Shaer, O. (2015). Eugenie: Multi-touch and tangible interaction for bio-design. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '15, pages 217–224, New York, NY, USA. ACM.
- Grubert, J., Morrison, A., Munz, H., and Reitmayr, G. (2012). Playing it real: magic lens and static peephole interfaces for games in a public space. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 231–240, New York, NY, USA. ACM.
- Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. *The Journal of Motor Behavior*, 19(4):486–517.
- Haller, M., Brandl, P., Leithinger, D., Leitner, J., Seifried, T., and Billinghurst, M. (2006). Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In Pan, Z., Cheok, A., Haller, M., Lau, R., Saito, H., and Liang, R., editors, *Advances in Artificial Reality and Tele-Existence*, volume 4282 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin Heidelberg.
- Haller, M., Forlines, C., Koeffel, C., Leitner, J., and Shen, C. (2010). Tabletop games: Platforms, experimental games and design recommendations. In *Art and Technology of Entertainment Computing and Communication*, pages 271–297. Springer.
- Han, J. Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 115–118, New York, NY, USA. ACM.
- Harrick, N. J. and Carlson, A. I. (1971). Internal reflection spectroscopy: Validity of effective thickness equations. *Appl. Opt.*, 10(1):19–23.
- Haubner, N., Schwanecke, U., Dörner, R., Lehmann, S., and Luderschmidt, J. (2013). Detecting interaction above digital tabletops using a single depth camera. *Machine Vision and Applications*, 24(8):1575–1587.
- He, D., Fuhu, D. H., Pape, D., Dawe, G., and S, D. (2000). Video-based measurement of system latency. In *International Immersive Projection Technology Workshop*.
- Henze, N. and Boll, S. (2010). Evaluation of an off-screen visualization for magic lens and dynamic peephole interfaces. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 191–194, New York, NY, USA. ACM.

- Herlihy, M. (1993). A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770.
- Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., and Butz, A. (2009). Interactions in the air: Adding further depth to interactive tabletops. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 139–148, New York, NY, USA. ACM.
- Hilliges, O., Kim, D., and Izadi, S. (2008). Creating malleable interactive surfaces using liquid displacement sensing. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 157–160.
- Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. (2010). Pen + touch = new tools. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 27–36, New York, NY, USA. ACM.
- Hodges, S., Izadi, S., Butler, A., Rustemi, A., and Buxton, B. (2007). Thinsight: Versatile multi-touch sensing for thin form-factor displays. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 259–268, New York, NY, USA. ACM.
- Hofer, R., Naeff, D., and Kunz, A. (2009). Flatir: Ftir multi-touch detection on a discrete distributed sensor array. In *Proceedings of the International Conference on Tangible and Embedded Interaction*, TEI '09, pages 317–322, New York, NY, USA. ACM.
- Holmquist, L. E., Sanneblad, J., and Gaye, L. (2003). Total recall: in-place viewing of captured whiteboard annotations. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pages 980–981, New York, NY, USA. ACM.
- Holz, C. and Baudisch, P. (2010). The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 581–590, New York, NY, USA. ACM.
- Holz, C. and Baudisch, P. (2013). Fiberio: A touchscreen that senses fingerprints. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 41–50, New York, NY, USA. ACM.
- Hung, Y.-P., Yang, Y.-S., Chen, Y.-S., Hsieh, I.-B., and Fuh, C.-S. (1998). Free-hand pointer by use of an active stereo vision system. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1244–1246 vol.2.
- Iacolina, S. A., Soro, A., and Scateni, R. (2011). Improving ftir based multi-touch sensors with ir shadow tracking. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '11, pages 241–246, New York, NY, USA. ACM.
- Ishii, H. (2008). Tangible bits: Beyond pixels. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction*, TEI '08, pages xv–xxv, New York, NY, USA. ACM.
- Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 234–241, New York, NY, USA. ACM.
- Jackson, D., Bartindale, T., and Olivier, P. (2009). Fiberboard: Compact multi-touch display using channeled light. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 25–28, New York, NY, USA. ACM.
- Jain, A., Klinker, L., Kranz, M., Stoeger, C., Blank, D., and Moesenlechner, L. (2006). Sketch-A-Move - Design Inspired Technology for Children.
- Jennings, C. (1999). Robust finger tracking with multiple cameras. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*, pages 152–160.
- Jetter, H.-C. (2013). *Design and Implementation of Post-WIMP Interactive Spaces with the ZOIL Paradigm*. PhD thesis.

- Jetter, H.-C., Dachsel, R., Reiterer, H., Quigley, A., Benyon, D., and Haller, M. (2013). Blended interaction: envisioning future collaborative interactive spaces. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 3271–3274, New York, NY, USA. ACM.
- Jetter, H.-C., Leifert, S., Gerken, J., Schubert, S., and Reiterer, H. (2012). Does (multi-)touch aid users' spatial memory and navigation in 'panning' and in 'zooming & panning' uis? In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 83–90, New York, NY, USA. ACM.
- Jin, Y., Dang, C. T., Prehofer, C., and André, E. (2014). A multi-display system for deploying and controlling home automation. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 399–402, New York, NY, USA. ACM.
- Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. (2007). The reactable: Exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 139–146, New York, NY, USA. ACM.
- Jota, R., Ng, A., Dietz, P., and Wigdor, D. (2013). How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2291–2300, New York, NY, USA. ACM.
- Kaltenbrunner, M. (2009). reactivation and tuio: A tangible tabletop toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 9–16, New York, NY, USA. ACM.
- Kaufmann, B. and Ahlström, D. (2013). Studying spatial memory and map navigation performance on projector phones with peephole interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3173–3176, New York, NY, USA. ACM.
- Keller, C., Bluteau, J., Blanch, R., and Coquillart, S. (2012). Pseudoweight: Making tabletop interaction with virtual objects more tangible. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, pages 201–204, New York, NY, USA. ACM.
- Kim, D., Dunphy, P., Briggs, P., Hook, J., Nicholson, J. W., Nicholson, J., and Olivier, P. (2010). Multi-touch authentication on tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1093–1102, New York, NY, USA. ACM.
- Kirton, T., Ogawa, H., Sommerer, C., and Mignonneau, L. (2008). Pins: a prototype model towards the definition of surface games. In *Proceedings of the ACM international conference on Multimedia*, MM '08, pages 953–956, New York, NY, USA. ACM.
- Klinkhammer, D., Tennie, J. O., Erdoes, P., and Reiterer, H. (2013). Body panning: A movement-based navigation technique for large interactive surfaces. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 37–40, New York, NY, USA. ACM.
- Klokmoose, C. N. and Beaudouin-Lafon, M. (2009). Vigo: Instrumental interaction in multi-surface environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 869–878, New York, NY, USA. ACM.
- Kojima, M., Sugimoto, M., Nakamura, A., Tomita, M., Inami, M., and Nii, H. (2006). Augmented coliseum: An augmented game environment with small vehicles. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 3–8, Washington, DC, USA. IEEE Computer Society.
- Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. (1985). Videoplac—an artificial reality. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '85, pages 35–40, New York, NY, USA. ACM.
- Langner, R., Augsburg, A., and Dachsel, R. (2014). Cubequery: Tangible interface for creating and manipulating database queries. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 423–426, New York, NY, USA. ACM.

- Leifert, S. (2011). The influence of grids on spatial and content memory. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 941–946, New York, NY, USA. ACM.
- Leitner, J., Haller, M., Yun, K., Woo, W., Sugimoto, M., Inami, M., Cheok, A. D., and Been-Lirn, H. D. (2010). Physical interfaces for tabletop games. *Comput. Entertain.*, 7:61:1–61:21.
- Leitner, J., Koeffel, C., and Haller, M. (2009). Bridging the gap between real and virtual objects for tabletop games. *International Journal*, 7(4):1–5.
- Lindley, S. E., Le Couteur, J., and Berthouze, N. L. (2008). Stirring up experience through movement in game play: effects on engagement and social behaviour. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 511–514, New York, NY, USA. ACM.
- MacKenzie, I. S. and Ware, C. (1993). Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 488–493, New York, NY, USA. ACM.
- MacLean, A., Young, R. M., Bellotti, V. M. E., and Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Hum.-Comput. Interact.*, 6(3):201–250.
- Magerkurth, C., Memisoglu, M., Engelke, T., and Streitz, N. (2004). Towards the next generation of tabletop gaming experiences. In *Proceedings of Graphics Interface 2004*, GI '04, pages 73–80, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.
- Malik, S. and Laszlo, J. (2004). Visual touchpad: A two-handed gestural input device. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ICMI '04, pages 289–296, New York, NY, USA. ACM.
- Marquardt, N., Kiemer, J., and Greenberg, S. (2010). What caused that touch? expressive interaction with a surface through fiduciary-tagged gloves. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 139–142, New York, NY, USA. ACM.
- Marquardt, N., Nacenta, M. A., Young, J. E., Carpendale, S., Greenberg, S., and Sharlin, E. (2009). The haptic tabletop puck: Tactile feedback for interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 85–92, New York, NY, USA. ACM.
- Martínez, R., Collins, A., Kay, J., and Yacef, K. (2011). Who did what? who said that?: Collaid: An environment for capturing traces of collaborative learning at the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 172–181, New York, NY, USA. ACM.
- Martinez-Maldonado, R., Dimitriadis, Y., Martinez-Monés, A., Kay, J., and Yacef, K. (2013). Capturing and analyzing verbal and physical collaborative learning interactions at an enriched interactive tabletop. *International Journal of Computer-Supported Collaborative Learning*, 8(4):455–485.
- Martinez Maldonado, R., Kay, J., Yacef, K., and Schwendimann, B. (2012). An interactive teacher's dashboard for monitoring groups in a multi-tabletop learning environment. In Cerri, S., Clancey, W., Papadourakis, G., and Panourgia, K., editors, *Intelligent Tutoring Systems*, volume 7315 of *Lecture Notes in Computer Science*, pages 482–492. Springer Berlin Heidelberg.
- Masoodian, M., Luz, S., and Kavenga, D. (2016). Nu-view: A visualization system for collaborative co-located analysis of geospatial disease data. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '16, pages 48:1–48:10, New York, NY, USA. ACM.
- Matejka, J., Grossman, T., Lo, J., and Fitzmaurice, G. (2009). The design and evaluation of multi-finger mouse emulation techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1073–1082, New York, NY, USA. ACM.
- Maybury, M. T. and Wahlster, W., editors (1998). *Readings in Intelligent User Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

- McAdam, C. and Brewster, S. (2011). Multimodal feedback for tabletop interactions. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS '11*, pages 274–275, New York, NY, USA. ACM.
- McGrath, W., Bowman, B., McCallum, D., Hincapié-Ramos, J. D., Elmqvist, N., and Irani, P. (2012). Branch-explore-merge: facilitating real-time revision control in collaborative visual exploration. In *Proceedings of the International conference on Interactive tabletops and surfaces, ITS '12*, pages 235–244, New York, NY, USA. ACM.
- Mehlmann, G., Häring, M., Janowski, K., Baur, T., Gebhard, P., and André, E. (2014). Exploring a model of gaze for grounding in multimodal hri. In *Proceedings of the 16th International Conference on Multimodal Interaction, ICMI '14*, pages 247–254, New York, NY, USA. ACM.
- Mi, H. and Sugimoto, M. (2011a). Design, implementations and applications of bidirectional tangible interfaces in a tabletop computing platform. In *Proceedings of the 2011 ACM Symposium on The Role of Design in UbiComp Research and Practice, RDURP '11*, pages 15–18, New York, NY, USA. ACM.
- Mi, H. and Sugimoto, M. (2011b). Hats: Interact using height-adjustable tangibles in tabletop interfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS '11*, pages 71–74, New York, NY, USA. ACM.
- Microsoft (2011). Microsoft pixelsense, samsung sur40.
- Miller, R. B. (1968). Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I, AFIPS '68 (Fall, part I)*, pages 267–277, New York, NY, USA. ACM.
- Moen, J. (2006). *KinAesthetic Movement Interaction : Designing for the Pleasure of Motion*. PhD thesis, University of Stockholm.
- Moscovich, T. and Hughes, J. F. (2008). Indirect mappings of multi-touch input using one and two hands. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08*, pages 1275–1284, New York, NY, USA. ACM.
- Müller-Tomfelde, C. (2010). *Tabletops - Horizontal Interactive Displays*. Springer Publishing Company, Incorporated, 1st edition.
- Murugappan, S., Vinayak, Elmqvist, N., and Ramani, K. (2012). Extended multitouch: Recovering touch posture and differentiating users using a depth camera. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12*, pages 487–496, New York, NY, USA. ACM.
- Nacenta, M. (2009). *Cross-display object movement in multi-display environments*. PhD thesis, University of Saskatchewan, <http://library2.usask.ca/theses/available/etd-01062010-123426/>.
- Nacenta, M. A., Gutwin, C., Aliakseyeu, D., and Subramanian, S. (2009). There and back again: Cross-display object movement in multi-display environments. *Human-Computer Interaction*, 24(1-2):170–229.
- Ng, A., Lepinski, J., Wigdor, D., Sanders, S., and Dietz, P. (2012). Designing for low-latency direct-touch input. In *Proceedings of the ACM symposium on User interface software and technology, UIST '12*, pages 453–464, New York, NY, USA. ACM.
- Norman, D. A. (1999). Affordance, conventions, and design. *interactions*, 6(3):38–43.
- Nowacka, D., Ladha, K., Hammerla, N. Y., Jackson, D., Ladha, C., Rukzio, E., and Olivier, P. (2013). Touchbugs: Actuated tangibles on multi-touch tables. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13*, pages 759–762, New York, NY, USA. ACM.
- NUIGroup (2014a). Community core vision. <http://ccv.nuigroup.com>.
- NUIGroup (2014b). Touchlib, a multi-touch development kit. <http://www.nuigroup.com/touchlib>.

- OpenCV (2015). Opencv, open computer vision library. <http://sourceforge.net/projects/opencvlibrary>.
- Pangaro, G. and Ishii, H. (2003). The actuated workbench: Computer-controlled actuation in tabletop tangible interfaces. *ACM Trans. Graph.*, 22(3):699–699.
- Patten, J. and Ishii, H. (2007). Mechanical constraints as computational constraints in tabletop tangible interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 809–818, New York, NY, USA. ACM.
- Pedersen, E. W. and Hornbæk, K. (2011). Tangible bots: Interaction with active tangibles in tabletop interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2975–2984, New York, NY, USA. ACM.
- Pedersen, E. W. and Hornbæk, K. (2014). Expressive touch: Studying tapping force on tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 421–430, New York, NY, USA. ACM.
- Peltonen, P., Kurvinen, E., Salovaara, A., Jacucci, G., Ilmonen, T., Evans, J., Oulasvirta, A., and Saarikko, P. (2008). It's mine, don't touch!: Interactions at a large multi-touch display in a city centre. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1285–1294, New York, NY, USA. ACM.
- Pfeuffer, K., Alexander, J., and Gellersen, H. (2015). *Gaze+touch vs. Touch: What's the Trade-off When Using Gaze to Extend Touch to Remote Displays?*, pages 349–367. Springer International Publishing, Cham.
- Piper, A. and Hollan, J. (2009). Analyzing multimodal communication around a shared tabletop display. In Wagner, I., Tellioglu, H., Balka, E., Simone, C., and Ciolfi, L., editors, *ECSCW 2009*, pages 283–302. Springer London.
- Ponnekanti, S. R., Johanson, B., Kiciman, E., and Fox, A. (2003). Portability, extensibility and robustness in iros. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, pages 11–.
- Rädle, R., Jetter, H.-C., Butscher, S., and Reiterer, H. (2013). The effect of egocentric body movements on users' navigation performance and spatial memory in zoomable user interfaces. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 23–32, New York, NY, USA. ACM.
- Rädle, R., Jetter, H.-C., Marquardt, N., Reiterer, H., and Rogers, Y. (2014). Huddlelamp: Spatially-aware mobile displays for ad-hoc around-the-table collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 45–54, New York, NY, USA. ACM.
- Ramakers, R., Vanacken, D., Luyten, K., Coninx, K., and Schöning, J. (2012). Carpus: A non-intrusive user identification technique for interactive surfaces. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 35–44, New York, NY, USA. ACM.
- Ramos, G., Boulos, M., and Balakrishnan, R. (2004). Pressure widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 487–494, New York, NY, USA. ACM.
- Rekimoto, J. (1997). Pick-and-drop: A direct manipulation technique for multiple computer environments. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, pages 31–39, New York, NY, USA. ACM.
- Rekimoto, J. and Saitoh, M. (1999). Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 378–385, New York, NY, USA. ACM.
- Ressler, S., Antonishek, B., Wang, Q., and Godil, A. (2001). Integrating active tangible devices with a synthetic environment for collaborative engineering. In *Proceedings of the Sixth International Conference on 3D Web Technology*, Web3D '01, pages 93–100, New York, NY, USA. ACM.

- Richter, J., Thomas, B. H., Sugimoto, M., and Inami, M. (2007). Remote active tangible interactions. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 39–42, New York, NY, USA. ACM.
- Riedenklaus, E., Hermann, T., and Ritter, H. (2012). An integrated multi-modal actuated tangible user interface for distributed collaborative planning. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, pages 169–174, New York, NY, USA. ACM.
- Robert, D., Wistorrt, R., Gray, J., and Breazeal, C. (2011). Exploring mixed reality robot gaming. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 125–128, New York, NY, USA. ACM.
- Rogers, Y. and Lindley, S. (2004). Collaborating around vertical and horizontal large interactive displays: which way is best? *Interacting with Computers*, 16(6):1133 – 1152.
- Rohs, M., Essl, G., Schöning, J., Naumann, A., Schleicher, R., and Krüger, A. (2009). Impact of item density on magic lens interactions. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 38:1–38:4, New York, NY, USA. ACM.
- Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., and Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83.
- Rosenfeld, D., Zawadzki, M., Sudol, J., and Perlin, K. (2004). Physical objects as bidirectional user interface elements. *IEEE Comput. Graph. Appl.*, 24(1):44–49.
- Roudaut, A., Lecolinet, E., and Guiard, Y. (2009). Microrolls: Expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 927–936, New York, NY, USA. ACM.
- Schmidt, D., Seifert, J., Rukzio, E., and Gellersen, H. (2012). A cross-device interaction style for mobiles and surfaces. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, pages 318–327, New York, NY, USA. ACM.
- Schmidt, H., Dang, C. T., Gessler, S., and Hauck, F. J. (2009). Model-driven development of adaptive applications with self-adaptive mobile processes. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*, OTM '09, pages 726–743, Berlin, Heidelberg. Springer-Verlag.
- Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., Motamedi, N., Müller, L., Olivier, P., Roth, T., and Von Zadow, U. (2008). Multi-Touch Surfaces: A Technical Guide.
- Schöning, J., Krüger, A., and Olivier, P. (2009). Multi-touch is dead, long live multi-touch. In *CHI 2009: Workshop on Multitouch and Surface Computing. ACM International Conference on Human Factors in Computing Systems (CHI-2009), April 4-9, Boston, MA, USA*, pages 7–11. ACM.
- Scott, S. D., Carpendale, M. S. T., and Inkpen, K. M. (2004). Territoriality in collaborative tabletop workspaces. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 294–303, New York, NY, USA. ACM.
- Scott, S. D. and Carpendale, S. (2006). Guest editors' introduction: Interacting with digital tabletops. *IEEE Comput. Graph. Appl.*, 26(5):24–27.
- Seyed, T., Burns, C., Costa Sousa, M., and Maurer, F. (2013). From small screens to big displays: understanding interaction in multi-display environments. In *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, IUI '13 Companion, pages 33–36, New York, NY, USA. ACM.
- Seyed, T., Burns, C., Costa Sousa, M., Maurer, F., and Tang, A. (2012). Eliciting usable gestures for multi-display environments. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, pages 41–50, New York, NY, USA. ACM.

- Sharma, A., Madhvanath, S., Shekhawat, A., and Billingham, M. (2011). Mozart: A multimodal interface for conceptual 3d modeling. In *Proceedings of the 13th International Conference on Multimodal Interfaces, ICMI '11*, pages 307–310, New York, NY, USA. ACM.
- Shen, C., Esenther, A., Forlines, C., and Ryall, K. (2006). Three modes of multisurface interaction and visualization. In *Information Visualization and Interaction Techniques for Collaboration across Multiple Displays Workshop associated with CHI*, volume 6.
- Siek, K. A., Rogers, Y., and Connelly, K. H. (2005). Fat finger worries: How older and younger users physically interact with pdas. In *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction, INTERACT'05*, pages 267–280, Berlin, Heidelberg. Springer-Verlag.
- Sousa, C. and Matsumoto, M. (2007). Study on fluent interaction with multi-touch in traditional gui environments. In *TENCON 2007 - 2007 IEEE Region 10 Conference*, pages 1–4.
- Speelpenning, T., Antle, A. N., Doering, T., and van den Hoven, E. (2011). Exploring how tangible tools enable collaboration in a multi-touch tabletop game. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part II, INTERACT'11*, pages 605–621, Berlin, Heidelberg. Springer.
- Spindler, M., Büschel, W., Winkler, C., and Dachsel, R. (2014). Tangible displays for the masses: Spatial interaction with handheld displays by using consumer depth cameras. *Personal Ubiquitous Comput.*, 18(5):1213–1225.
- Spindler, M., Stellmach, S., and Dachsel, R. (2009). Paperlens: advanced magic lens interaction above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS '09*, pages 69–76, New York, NY, USA. ACM.
- Sugimoto, M., Kodama, K., Nakamura, A., Kojima, M., and Inami, M. (2007). A display-based tracking system: Display-based computing for measurement systems. In *Proceedings of the 17th International Conference on Artificial Reality and Telexistence, ICAT '07*, pages 31–38, Washington, DC, USA. IEEE Computer Society.
- Takeoka, Y., Miyaki, T., and Rekimoto, J. (2010). Z-touch: An infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS '10*, pages 91–94, New York, NY, USA. ACM.
- Takeuchi, Y. and Sugimoto, M. (2012). An immersive surface for 3d interactions. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces, ITS '12*, pages 359–362, New York, NY, USA. ACM.
- Tan, D. S., Pausch, R., Stefanucci, J. K., and Proffitt, D. R. (2002). Kinesthetic cues aid spatial memory. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems, CHI EA '02*, pages 806–807, New York, NY, USA. ACM.
- Tandler, P. (2000). Architecture of beach: The software infrastructure for roomware environments. In *ACM Conference On Computer supported Cooperative Work (CSCW 2000)*, pages 2–6.
- Tanev, I., Joachimczak, M., and Shimohara, K. (2006). Evolution of driving agent, remotely operating a scale model of a car with obstacle avoidance capabilities. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation, GECCO '06*, pages 1785–1792, New York, NY, USA. ACM.
- Tanev, I. and Shimohara, K. (2007). Towards human competitive driving of scale model of a car. *SIGEVOlution*, 2:14–26.
- Terrenghi, L., Kirk, D., Sellen, A., and Izadi, S. (2007). Affordances for manipulation of physical versus digital media on interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07*, pages 1157–1166, New York, NY, USA. ACM.
- Tianding, C. (2008). A solution of computer vision based real-time hand pointing recognition. In *Control Conference, 2008. CCC 2008. 27th Chinese*, pages 384–388.

- Tsao, C.-L., Kakumanu, S., and Sivakumar, R. (2011). Smartvnc: an effective remote computing solution for smartphones. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, MobiCom '11, pages 13–24, New York, NY, USA. ACM.
- Tse, E., Greenberg, S., Shen, C., and Forlines, C. (2007). Multimodal multiplayer tabletop gaming. *Comput. Entertain.*, 5(2).
- Tse, E., Shen, C., Greenberg, S., and Forlines, C. (2006). Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 336–343, New York, NY, USA. ACM.
- Tse, E. H. (2007). *Multimodal Co-located Interaction*. PhD thesis, University of Calgary.
- Tuddenham, P., Kirk, D., and Izadi, S. (2010). Graspables revisited: Multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2223–2232, New York, NY, USA. ACM.
- Turk, M. (2014). Review article: Multimodal interaction: A review. *Pattern Recogn. Lett.*, 36:189–195.
- Ullmer, B. and Ishii, H. (2000). Emerging frameworks for tangible user interfaces. *IBM Syst. J.*, 39(3-4):915–931.
- Underkoffler, J. and Ishii, H. (1998). Illuminating light: An optical design tool with a luminous-tangible interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, pages 542–549, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- Underkoffler, J. and Ishii, H. (1999). Urp: A luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 386–393, New York, NY, USA. ACM.
- Valdes, C., Eastman, D., Grote, C., Thatte, S., Shaer, O., Mazalek, A., Ullmer, B., and Konkel, M. K. (2014). Exploring the design space of gestural interaction with active tokens through user-defined gestures. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 4107–4116, New York, NY, USA. ACM.
- Valkov, D., Steinicke, F., Bruder, G., and Hinrichs, K. (2011). 2d touching of 3d stereoscopic objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1353–1362, New York, NY, USA. ACM.
- van Asselen, M. (2005). *Neurocognition of spatial memory : Studies in patients with acquired brain damage and healthy participants*. PhD thesis, Universiteit Utrecht.
- van de Camp, F. and Stiefelhagen, R. (2013). Gluetk: A framework for multi-modal, multi-display human-machine-interaction. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 329–338, New York, NY, USA. ACM.
- Vidal, M. (2014). *Exploring Eye Movements for Natural Human-Computer Interaction*. PhD thesis, Lancaster University.
- Voelker, S., Cherek, C., Thar, J., Karrer, T., Thoresen, C., vergard, K. I., and Borchers, J. (2015a). Percs: Persistently trackable tangibles on capacitive multi-touch displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pages 351–356, New York, NY, USA. ACM.
- Voelker, S., Matvienko, A., Schöning, J., and Borchers, J. (2015b). Combining direct and indirect touch input for interactive workspaces using gaze input. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, SUI '15, pages 79–88, New York, NY, USA. ACM.
- Vogel, D. and Balakrishnan, R. (2010). Occlusion-aware interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 263–272, New York, NY, USA. ACM.
- Vogel, D. and Baudisch, P. (2007). Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 657–666, New York, NY, USA. ACM.

- Voida, S., Tobiasz, M., Stromer, J., Isenberg, P., and Carpendale, S. (2009). Getting practical with interactive tabletop displays: designing for dense data, "fat fingers," diverse interactions, and face-to-face collaboration. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 109–116, New York, NY, USA. ACM.
- Wagner, J., Nancel, M., Gustafson, S. G., Huot, S., and Mackay, W. E. (2013). Body-centric design space for multi-surface interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1299–1308, New York, NY, USA. ACM.
- Wallace, J., Ha, V., Ziola, R., and Inkpen, K. (2006). Swordfish: User tailored workspaces in multi-display environments. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1487–1492, New York, NY, USA. ACM.
- Wang, F., Cao, X., Ren, X., and Irani, P. (2009). Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 23–32, New York, NY, USA. ACM.
- Wang, F. and Ren, X. (2009). Empirical evaluation for finger input properties in multi-touch interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1063–1072, New York, NY, USA. ACM.
- Ware, C. (2008). *Visual Thinking: For Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Wasinger, R. (2006). *Multimodal Interaction with Mobile Devices: Fusing a Broad Spectrum of Modality Combinations*. PhD thesis, University of Saarbruecken.
- Wechsung, I. (2014). *An Evaluation Framework for Multimodal Interaction: Determining Quality Aspects and Modality Choice*. Springer Publishing Company, Incorporated.
- Weiser, M. (1991). The computer for the 21st century. *Scientific American*, pages 94–104.
- Weiss, M., Schwarz, F., Jakubowski, S., and Borchers, J. (2010). Madgets: Actuating widgets on interactive tabletops. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 293–302, New York, NY, USA. ACM.
- Weiss, M., Wagner, J., Jennings, R., Jansen, Y., Khoshabeh, R., Hollan, J. D., and Borchers, J. (2009). Slapbook: Tangible widgets on multi-touch tables in groupware environments. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, TEI '09, pages 297–300, New York, NY, USA. ACM.
- Wellner, P. (1993). Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96.
- Wellner, P., Mackay, W., and Gold, R. (1993). Back to the real world. *Commun. ACM*, 36(7):24–26.
- Wigdor, D., Penn, G., Ryall, K., Esenther, A., and Shen, C. (2007). Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, October 10-12 2007, Newport, Rhode Island, USA, pages 60–67.
- Wigdor, D., Shen, C., Forlines, C., and Balakrishnan, R. (2006). Table-centric interactive spaces for real-time collaboration. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 103–107, New York, NY, USA. ACM.
- Wilson, A. D. (2010). Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA. ACM.
- Wilson, A. D. and Benko, H. (2010). Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 273–282, New York, NY, USA. ACM.
- Wu, A., Joyner, D., and Do, E. Y.-L. (2010). Move, beam, and check! imagineering tangible optical chess on an interactive tabletop display. *Comput. Entertain.*, 8:20:1–20:15.

- Wu, A., Yim, J.-B., Caspary, E., Mazalek, A., Chandrasekharan, S., and Nersessian, N. J. (2011). Kinesthetic pathways: A tabletop visualization to support discovery in systems biology. In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, CaC '11, pages 21–30, New York, NY, USA. ACM.
- Xiao, R., Schwarz, J., and Harrison, C. (2015). Estimating 3d finger angle on commodity touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops and Surfaces*, ITS '15, pages 47–50, New York, NY, USA. ACM.
- Yamamoto, M., Komeda, M., Nagamatsu, T., and Watanabe, T. (2011). Hyakunin-eyesshu: A tabletop hyakunin-issu game with computer opponent by the action prediction based on gaze detection. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, pages 5:1–5:4, New York, NY, USA. ACM.
- Yao, J., Fernando, T., Tawfik, H., Armitage, R., and Billing, I. (2006). Towards a collaborative urban planning environment. In Shen, W.-m., Chao, K.-M., Lin, Z., Barthès, J.-P., and James, A., editors, *Computer Supported Cooperative Work in Design II*, volume 3865 of *Lecture Notes in Computer Science*, pages 554–562. Springer Berlin Heidelberg.
- Yee, K.-P. (2003). Peephole displays: pen interaction on spatially aware handheld computers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 1–8, New York, NY, USA. ACM.
- Yee, W. (2009). Potential limitations of multi-touch gesture vocabulary: Differentiation, adoption, fatigue. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, pages 291–300, Berlin, Heidelberg. Springer-Verlag.
- Zhu, S., Yu, A., Hawley, D., and Roy, R. (1986). Frustrated total internal reflection: a demonstration and review. *Am. J. Phys.*, 54(7):601–607.

APPENDIX

A.1 Source code for Atmel ATmega8 pulse controller.

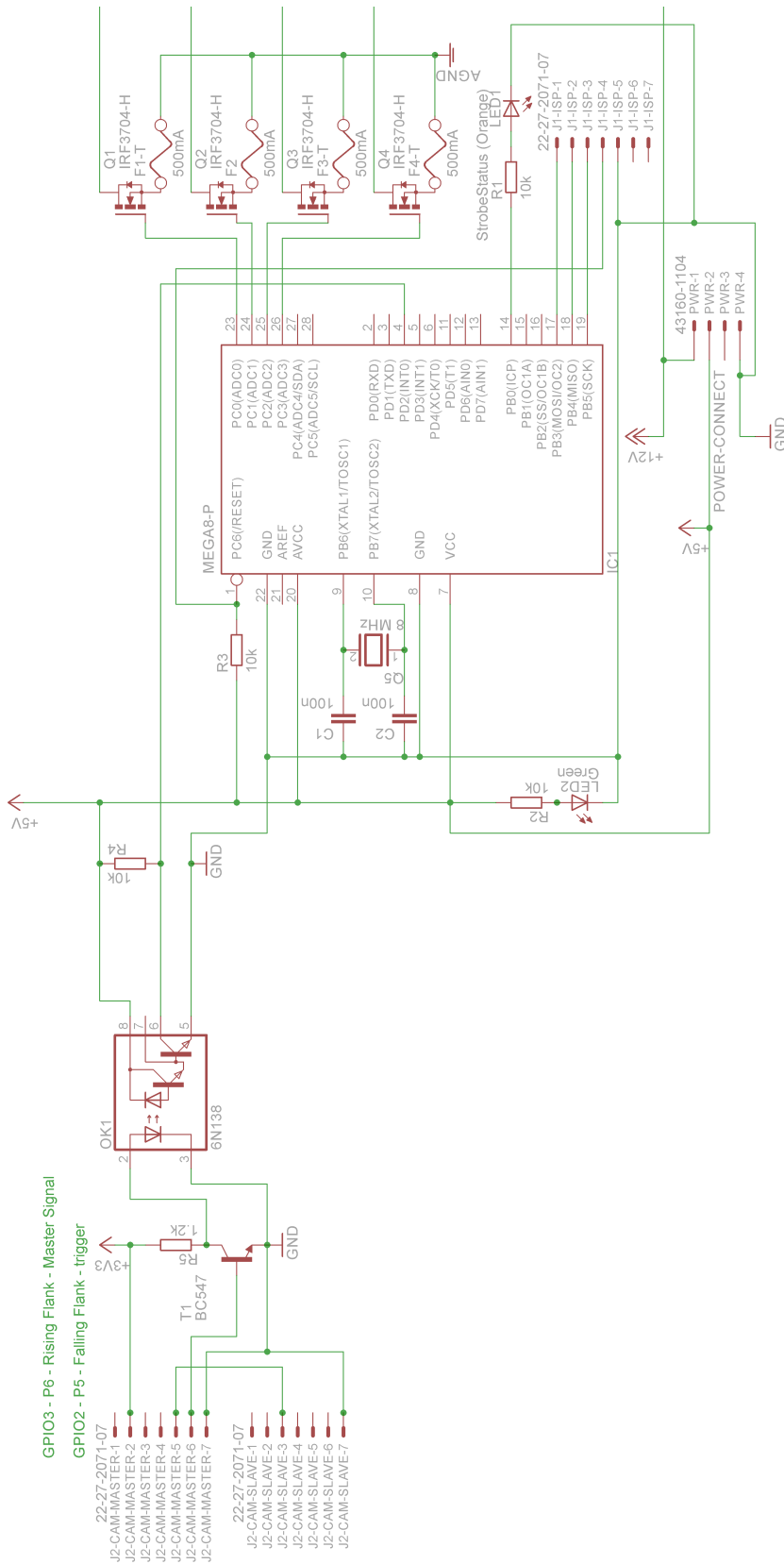
```
1  /*
2  StrobeFlash for AT90S8535
3  Autor:      Chi-Tai Dang (dang@informatik.uni-augsburg.de)
4  Date:      01/31/2009
5  Clock: 8 MHz -> 1 Clockcycle lasts 0.123us - 125ns -> 32us
6
7  Camera lowest = 100us - 0.0001s
8  -> 3 * 32us      // tp
9  -> 621 * 32us    // tr
10
11  D      =0.005          // tp / T
12  tp     =0.0001 s
13  tr     =0.0199s
14  T      =0.02 s        // tp + tr
15  fps(max) = 50
16  */
17
18  #include "avr/io.h"
19  #include "avr/interrupt.h"
20  #include "avr/sleep.h"
21
22  // #define TEST_MODE
23
24  int nTp;
25  int nTr;
26  int nRelease;
27  int nTimer;
28  int nInPulse;
29
30  SIGNAL ( SIG_OVERFLOW0 )
31  {
32      nTimer++;
33
34      if ( nInPulse == 0 ) {
35          // Count Release for indicator led
36          if ( nTimer >= nRelease ) {
37              // Disable timer 0
38              TIMSK0 = 0;
39              // Clear indicator and signal
40              PORTB = PORTB & 0xFA;
41              // Initialize timer counter to 0
42              TCNT0 = 0;
43          }
44          return;
45      }
46
47      if ( nTimer >= nTp ) {
48          // Clear signal
49          PORTB = PORTB & 0xFB;
50          if ( nTimer >= nTr ) {
51              // We're now leaving the pulse-sequence
52              nInPulse = 0;
53          }
54      }
55  }
```

```

56
57
58 SIGNAL ( SIG_INTERRUPT0 )
59 {
60     // If we're within a pulse-sequence, then return immediately
61     if ( nInPulse )
62         return;
63
64     // Disable timer 0
65     TIMSK0 = 0;
66     // We're now beginning a pulse-sequence
67     nInPulse = 1;
68     nTimer = 0;
69     // Initialize timer counter to 0
70     TCNT0 = 0;
71     // Enable timer 0
72     TIMSK0 = 2;
73     // Rise signal and indicator
74     PORTB = PORTB | 0x5;
75 }
76
77
78 int main ( )
79 {
80     // Set directions for port B (LEDs) to output
81     DDRB = 0xFD;
82     PORTB = 0;
83     // Pulse values for 8MHz
84 #ifndef TEST_MODE
85     nTp = 100;
86     nTr = 100 + nTp;
87 #else
88     nTp = 3;
89     nTr = 621 + nTp;
90 #endif
91     nRelease = 3 * nTr;
92     nTimer = 0;
93     nInPulse = 0;
94
95     // Enable INT0/INT1 on rising flank
96     //MCUCR |= 0x3;
97     MCUCR |= 0xF;
98     // Allow INT0 (extern)
99     //GICR = 1 << INTF0;
100     GIMSK = 1 << INTF0;
101
102 #ifndef TEST_MODE
103     TCCR0B = 5; // prescaler: 5 ~ 1024 (7.8125 kHz)
104 #else
105     TCCR0B = 1; // prescaler: 1 = 1
106 #endif
107
108     // Initialize timer counter to 0
109     TCNT0 = 0;
110     sei ( ); // enable interrupts
111     set_sleep_mode ( SLEEP_MODE_IDLE );
112
113     while ( 1 ) {
114         sleep_mode ( );
115     }
116 }

```

A.2 Schematic Diagram of the Controller-Board for Pulsed Illumination.



A.4 Multi-Surface Environments Review-Literature (sorted by index)

P1 : Ajaj, Jacquemin, Vernier, RVDT - A Design Space for Multiple Input Devices, Multiple Views and Multiple Display Surfaces Combination.

<http://dx.doi.org/10.1145/1647314.1647372>

P2 : Abad et al., Multi-Surface Interactions with Geospatial Data - A Systematic Review.

<http://dx.doi.org/10.1145/2669485.2669505>

P3 : Adachi et al., Forearm menu - using forearm as menu widget on tabletop system.

<http://dx.doi.org/10.1145/2512349.2512393>

P4 : Beaudouin-Lafon, Lessons Learned from the WILD Room, a Multisurface Interactive Environment.

<http://dx.doi.org/10.1145/2044354.2044376>

P5 : Camp, Stiefelhausen, glueTK - A Framework For Multi-Modal, Multi-Display Human-Machine-Interaction.

<http://dx.doi.org/10.1145/2449396.2449440>

P6 : Chokshi et al., ePlan Multi-Surface - A Multi-Surface Environment for Emergency Response Planning Exercises.

<http://dx.doi.org/10.1145/2669485.2669520>

P7 : Coffey et al., Slice WIM - A Multi-Surface, Multi-Touch Interface for Overview Detail Exploration of Volume Datasets in Virtual Reality.

<http://dx.doi.org/10.1145/1944745.1944777>

P8 : Domova et al., An interactive surface solution to support collaborative work onboard ships.

<http://dx.doi.org/10.1145/2512349.2512804>

P9 : Echtler et al., Supporting Casual Interactions between Board Games on Public Tabletop Displays and Mobile Devices.

<http://dx.doi.org/10.1007/s00779-009-0246-3>

P10 : Forlines, Lilien, Adapting a Single-user, Single-display Molecular Visualization Application for Use in a Multi-user, Multi-Display Environment.

<http://dx.doi.org/10.1145/1385569.1385635>

P11 : Gjerlufsen et al., Shared Substance - Developing Flexible Multi-Surface Applications.

<http://dx.doi.org/10.1145/1978942.1979446>

P12 : Hartmann et al., HydraScope - Creating Multi-Surface Meta-Applications Through View Synchronization and Input Multiplexing.

<http://dx.doi.org/10.1145/2491568.2491578>

P13 : Lee et al., Gesture-Based Interface for Connection and Control of Multi-device in a Tabletop Display Environment.

http://dx.doi.org/10.1007/978-3-642-02577-8_24

P14 : Meissner, Eck, Extending the Fire Dispatch System into the Mobile Domain.

http://dx.doi.org/10.1007/978-3-540-75668-2_16

P15 : Peinsipp-Byma, Geisler, Bader, Digital Map and Situation Surface - A Team-oriented Multi-Display Workspace for Network Enabled Situation Analysis.

<http://dx.doi.org/10.1117/12.817903>

P16 : Pratte et al., Exploring Multi-Surface Interactions in Retail Environments.

<http://dx.doi.org/10.1145/2669485.2669552>

P17 : Spindler et al., Tangible Displays For The Masses - Spatial Interaction With Handheld Displays By Using Consumer Depth Cameras.

<http://dx.doi.org/10.1007/s00779-013-0730-7>

P18 : Schneider et al., WALDEN - Multi-Surface Multi-Touch Simulation of Climate Change and Species Loss in thoreaus Woods.

<http://dx.doi.org/10.1145/2396636.2396707>

P19 : Scott et al., Surface Ghosts - Promoting Awareness of Transferred Objects during Pick-and-Drop Transfer in Multi-Surface Environments.

<http://dx.doi.org/10.1145/2669485.2669508>

P20 : Seifert et al., MobiSurf - Improving Co-located Collaboration through Integrating Mobile Devices and Interactive Surfaces.

<http://dx.doi.org/10.1145/2396636.2396644>

- P21** : Seyed et al., SkyHunter - A Multi-Surface Environment for Supporting Oil and Gas Exploration.
<http://dx.doi.org/10.1145/2512349.2512798>
- P22** : Yao et al., Towards a Collaborative Urban Planning Environment.
http://dx.doi.org/10.1007/11686699_56
- P23** : Yoshimoto et al., Interaction Design of 2D-3D Map Navigation On Wall And Tabletop Displays.
<http://dx.doi.org/10.1145/2076354.2076402>
- P24** : Bachl et al., The Effects of Personal Displays and Transfer Techniques on Collaboration Strategies in Multi-touch Based Multi-Display Environments.
http://dx.doi.org/10.1007/978-3-642-23765-2_26
- P25** : Cheng, Li, Mueller-Tomfelde, Supporting Interaction and Collaboration on Large Displays using Tablet Devices.
<http://dx.doi.org/10.1145/2254556.2254708>
- P26** : Chung et al., VisPorter - Facilitating Information Sharing For Collaborative Sensemaking On Multiple Displays.
<http://dx.doi.org/10.1007/s00779-013-0727-2>
- P27** : Collomb et al., Extending Drag-and-drop to New Interactive Environments - a Multi-display, Multi-instrument and Multi-user Approach.
<http://dx.doi.org/10.1016/j.intcom.2008.07.004>
- P28** : Nacenta et al., There and Back Again - Cross-Display Object Movement in Multi-Display Environments.
<http://dx.doi.org/10.1080/07370020902819882>
- P29** : Dippon, Wiedermann, Klinker, Seamless integration of mobile devices into interactive surface environments.
<http://dx.doi.org/10.1145/2396636.2396693>
- P30** : Azazi et al., Using Multiple Kinects to Build Larger Multi-Surface Environments.
- P31** : Burns et al., A Usable API for Multi-Surface Systems.
<http://dx.doi.org/10.1145/2449396.2449440>
- P32** : Ponnekanti et al., Portability, Extensibility and Robustness in iROS.
<http://dx.doi.org/10.1109/PERCOM.2003.1192722acmid=826373>
- P33** : Roman et al., Gaia - A Middleware Infrastructure to Enable Active Spaces.
<http://dx.doi.org/10.1109/MPRV.2002.1158281>
- P34** : Sakurai et al., A Middleware for Seamless Use of Multiple Displays.
http://dx.doi.org/10.1007/978-3-540-70569-7_23
- P35** : Tandler, Architecture of BEACH - The Software Infrastructure for Roomware Environments.
- P36** : Wigdor et al., WeSpace - The Design, Development, and Deployment of a Walk-Up and Share Multi-Surface Collaboration System.
<http://dx.doi.org/10.1145/1518701.1518886>
- P37** : Lee, Kim, Kim, Gesture-Based Interactions on Multiple Large Displays with a Tabletop Interface.
http://dx.doi.org/10.1007/978-3-540-73281-5_102
- P38** : McGrath et al., Branch-Explore-Merge - Facilitating Real-Time Revision Control in Co-Located Collaborative Visual Exploration.
<http://dx.doi.org/10.1145/2396636.2396673>
- P39** : Nacenta et al., A Comparison of Techniques for Multi-Display Reaching.
<http://dx.doi.org/10.1145/1054972.1055024>
- P40** : Nacenta et al., E-conic - a Perspective-Aware Interface for Multi-Display Environments.
<http://dx.doi.org/10.1145/1294211.1294260>
- P41** : Piazza et al., Holy Smartphones and Tablets, Batman - Mobile Interactions Dynamic Duo.
<http://dx.doi.org/10.1145/2525194.2525205>
- P42** : Boring et al., Touch Projector - Mobile Interaction through Video.
<http://dx.doi.org/10.1145/1753326.1753671>
- P43** : Diez et al., Sharing Your View - A Distributed User Interface Approach For Reviewing Emergency Plans.
<http://dx.doi.org/10.1016/j.ijhcs.2013.04.008>
- P44** : Peinsipp-Byma et al., Situation Analysis At A Digital Situation Table With Fovea-Tablett.
<http://dx.doi.org/10.1117/12.704367>

P45 : Seyed et al., Eliciting Usable Gestures for Multi-Display Environments.
<http://dx.doi.org/10.1145/2396636.2396643>

P46 : Shen et al., Three Modes of Multi-Surface Interaction and Visualization.

P47 : Spindler, Bueschel, Dachzelt, Use Your Head - Tangible Windows for 3D Information Spaces in a Tabletop Environment.
<http://dx.doi.org/10.1145/2396636.2396674>

P48 : Spindler, Stellmach, Dachzelt, PaperLens Advanced Magic Lens Interaction Above the Tabletop.
<http://dx.doi.org/10.1145/1731903.1731920>

P49 : Streitz et al., i-LAND - An interactive Landscape for Creativity and Innovation.
<http://dx.doi.org/10.1145/302979.303010>

P50 : Wagner et al., Body-centric Design Space for Multi-surface Interaction.
<http://dx.doi.org/10.1145/2470654.2466170>

P51 : Wigdor et al., Table-Centric Interactive Spaces for Real-Time Collaboration.
<http://dx.doi.org/10.1145/1133265.1133286>

P52 : Prieto, Studying Teacher Cognitive Load in Multi-tabletop Classrooms Using Mobile Eye-tracking.
<http://dx.doi.org/10.1145/2669485.2669543>

P53 : Kaufmann et al., 3MF - A Service-Oriented Mobile Multimodal Interaction Framework.

P54 : Schmidt et al., A Cross-Device Interaction Style for Mobiles and Surfaces.
<http://dx.doi.org/10.1145/2317956.2318005>

P55 : AlAgha, Towards a Teacher-Centric Approach for Multi-Touch Surfaces in Classrooms.
<http://dx.doi.org/10.1145/1936652.1936688>

P56 : Rodrigues, Bancada - Using Mobile Zoomable Lenses for Geospatial Exploration.
<http://dx.doi.org/10.1145/2669485.2669555>

P57 : Spindler, Tangible Views for Information Visualization.
<http://dx.doi.org/10.1145/1936652.1936684>

P58 : Song, WYSIWYF - Exploring and Annotating Volume Data with a Tangible Handheld Device.
<http://dx.doi.org/10.1145/1978942.1979140>

P59 : Kim et al., Hugin - A Framework for Awareness and Coordination in Mixed-Presence Collaborative Information Visualization.
<http://dx.doi.org/10.1145/1936652.1936694>

P60 : Schmidt et al., Personal Clipboards for Individual Copy-and-Paste on Shared Multi-User Surfaces.
<http://dx.doi.org/10.1145/2470654.2466457>

P61 : Streitz et al., Roomware - Towards the next generation of human-computer interaction based on an integrated design of real and virtual worlds.

P62 : Schmidt et al., PhoneTouch a technique for direct phone interaction on surfaces.
<http://dx.doi.org/10.1145/1866029.1866034>

P63 : Olwal, Feiner, Spatially Aware Handhelds for High-Precision Tangible Interaction with Large Displays.
<http://dx.doi.org/10.1145/1517664.1517705>

P64 : Nacenta et al., Perspective Cursor - Perspective-Based Interaction For Multi-Display Environments.
<http://dx.doi.org/10.1145/1124772.1124817>

P65 : Badam et al., PolyChrome - A Cross-Device Framework for Collaborative Web Visualization.
<http://dx.doi.org/10.1145/2669485.2669518>

P66 : Houben et al., ActivitySpace - Managing Device Ecologies in an Activity-Centric Configuration Space.
<http://dx.doi.org/10.1145/2669485.2669493>

P67 : Doering et al., Exploring Gesture-Based Interaction Techniques in MultiDisplay Environments with Mobile Phones and a Multi-Touch Table.
<http://dx.doi.org/10.1145/1842993.1843097>

P68 : Maldonado et al., An Interactive Teachers Dashboard for Monitoring Groups in a Multi-tabletop Learning Environment.
http://dx.doi.org/10.1007/978-3-642-30950-2_62

- P69** : Kitamura et al., Multi-modal Interface in Multi-Display Environment for Multi-users.
http://dx.doi.org/10.1007/978-3-642-02577-8_8
- P70** : Jetter et al., Model-Based Design and Implementation of Interactive Spaces for Information Interaction.
http://dx.doi.org/10.1007/978-3-642-16488-0_3
- P71** : Jaing et al., System Design for the WeSpace - Linking Personal Devices to a Table-Centered Multi-User, Multi-Surface Environment.
<http://dx.doi.org/10.1109/TABLETOP.2008.4660191>
- P72** : Lee et al., Dual Interactions between Multi-display and Smartphone for Collaborative Design and Sharing.
<http://dx.doi.org/10.1109/VR.2011.5759478>
- P73** : Scott et al., Cross-Device Transfer in a Collaborative MultiSurface Environment without User Identification.
<http://dx.doi.org/10.1109/CTS.2014.6867568>
- P74** : Fukazawa et al., Comparison of Multimodal Interactions in Perspective-corrected Multi-display Environment.
<http://dx.doi.org/10.1109/3DUI.2010.5444711>
- P75** : Lapides et al., Seamless Mixed Reality Tracking in Tabletop Reservoir Engineering Interaction.
<http://dx.doi.org/10.1145/2254556.2254694>
- P76** : Bardram et al., The Design and Architecture of ReticularSpaces - an Activity-Based Computing Framework for Distributed and Collaborative SmartSpaces.
<http://dx.doi.org/10.1145/2305484.2305529>
- P77** : Bardram et al., ReticularSpaces - Activity-Based Computing Support for Physically Distributed and Collaborative Smart Spaces.
<http://dx.doi.org/10.1145/2207676.2208689>
- P78** : Seifert et al., From the private into the public - privacy-respecting mobile interaction techniques for sharing data on surfaces.
<http://dx.doi.org/10.1007/s00779-013-0667-x>
- P79** : Nacenta et al., The LunchTable - A Multi-User, Multi-Display System for Information Sharing in Casual Group Interactions.
<http://dx.doi.org/10.1145/2307798.2307816>
- P80** : Forlines et al., Multi-User, Multi-Display Interaction with a Single-User, Single-Display Geospatial Application.
<http://dx.doi.org/10.1145/1166253.1166296>
- P81** : Fei et al., Positioning Portals with Peripheral NFC Tags to Embody Trans-Surface Interaction.
<http://dx.doi.org/10.1145/2512349.2514593>
- P82** : Bader et al., Lift-and-Drop - Crossing Boundaries in a Multi-Display Environment by Airlift.
<http://dx.doi.org/10.1145/1842993.1843019>
- P83** : Wigdor et al., Perception of Elementary Graphical Elements in Tabletop and Multi-Surface Environments.
<http://dx.doi.org/10.1145/1240624.1240701>
- P84** : Oskamp et al., TerraGuide - Design and Evaluation of a Multi-Surface Environment for Terrain Visibility Analysis.
<http://dx.doi.org/10.1145/2702123.2702480>
- P85** : Maldonado et al., Multi-touch Technology in a Higher-Education Classroom - Lessons In-The-Wild.
<http://dx.doi.org/10.1145/2686612.2686647>
- P86** : Bier et al., Toolglass and magic lenses the see-through interface.
<http://dx.doi.org/10.1145/166117.166126>
- P87** : Kurdyukova et al., Studying User-defined iPad Gestures for Interaction in Multi-display Environment.
<http://dx.doi.org/10.1145/2166966.2166984>
- P88** : Heilig et al., Search, Explore and Navigate - Designing a Next Generation Knowledge Media Workbench.
- P89** : Johanson et al., PointRight - Experience with Flexible Input Redirection in Interactive Workspaces.
<http://dx.doi.org/10.1145/571985.572019>
- P90** : Luff et al., Flexible Ecologies and Incongruent Locations.
<http://dx.doi.org/10.1145/2702123.2702286>
- P91** : Reilly et al., Mapping out Work in a Mixed Reality Project Room.
<http://dx.doi.org/10.1145/2702123.2702506>

A.5 Multi-Surface Environments Review-Literature (sorted by authors).

Abad et al., Multi-Surface Interactions with Geospatial Data - A Systematic Review, P2, <http://dx.doi.org/10.1145/2669485.2669505>

Adachi et al., Forearm menu - using forearm as menu widget on tabletop system, P3, <http://dx.doi.org/10.1145/2512349.2512393>

Ajaj, Jacquemin, Vernier, RVDT - A Design Space for Multiple Input Devices, Multiple Views and Multiple Display Surfaces Combination, P1, <http://dx.doi.org/10.1145/1647314.1647372>

AlAgha, Towards a Teacher-Centric Approach for Multi-Touch Surfaces in Classrooms, P55, <http://dx.doi.org/10.1145/1936652.1936688>

Azazi et al., Using Multiple Kinects to Build Larger Multi-Surface Environments, P30.

Bachl et al., The Effects of Personal Displays and Transfer Techniques on Collaboration Strategies in Multi-touch Based Multi-Display Environments, P24, http://dx.doi.org/10.1007/978-3-642-23765-2_26

Badam et al., PolyChrome - A Cross-Device Framework for Collaborative Web Visualization, P65, <http://dx.doi.org/10.1145/2669485.2669518>

Bader et al., Lift-and-Drop - Crossing Boundaries in a Multi-Display Environment by Airlift, P82, <http://dx.doi.org/10.1145/1842993.1843019>

Bardram et al., The Design and Architecture of ReticularSpaces - an Activity-Based Computing Framework for Distributed and Collaborative SmartSpaces, P76, <http://dx.doi.org/10.1145/2305484.2305529>

Bardram et al., ReticularSpaces - Activity-Based Computing Support for Physically Distributed and Collaborative Smart Spaces, P77, <http://dx.doi.org/10.1145/2207676.2208689>

Beaudouin-Lafon, Lessons Learned from the WILD Room, a Multisurface Interactive Environment, P4, <http://dx.doi.org/10.1145/2044354.2044376>

Bier et al., Toolglass and magic lenses the see-through interface, P86, <http://dx.doi.org/10.1145/166117.166126>

Boring et al., Touch Projector - Mobile Interaction through Video, P42, <http://dx.doi.org/10.1145/1753326.1753671>

Burns et al., A Usable API for Multi-Surface Systems, P31, <http://dx.doi.org/10.1145/2449396.2449440>

Camp, Stiefelhagen, glueTK - A Framework For Multi-Modal, Multi-Display Human-Machine-Interaction, P5, <http://dx.doi.org/10.1145/2449396.2449440>

Cheng, Li, Mueller-Tomfelde, Supporting Interaction and Collaboration on Large Displays using Tablet Devices, P25, <http://dx.doi.org/10.1145/2254556.2254708>

Chokshi et al., ePlan Multi-Surface - A Multi-Surface Environment for Emergency Response Planning Exercises, P6, <http://dx.doi.org/10.1145/2669485.2669520>

Chung et al., VisPorter - Facilitating Information Sharing For Collaborative Sensemaking On Multiple Displays, P26, <http://dx.doi.org/10.1007/s00779-013-0727-2>

Coffey et al., Slice WIM - A Multi-Surface, Multi-Touch Interface for Overview Detail Exploration of Volume Datasets in Virtual Reality, P7, <http://dx.doi.org/10.1145/1944745.1944777>

Collomb et al., Extending Drag-and-drop to New Interactive Environments - a Multi-display, Multi-instrument and Multi-user Approach, P27, <http://dx.doi.org/10.1016/j.intcom.2008.07.004>

Diez et al., Sharing Your View - A Distributed User Interface Approach For Reviewing Emergency Plans, P43, <http://dx.doi.org/10.1016/j.ijhcs.2013.04.008>

Dippon, Wiedermann, Klinker, Seamless integration of mobile devices into interactive surface environments, P29, <http://dx.doi.org/10.1145/2396636.2396693>

Doering et al., Exploring Gesture-Based Interaction Techniques in MultiDisplay Environments with Mobile Phones and a Multi-Touch Table, P67, <http://dx.doi.org/10.1145/1842993.1843097>

- Domova et al.**, An interactive surface solution to support collaborative work onboard ships, P8, <http://dx.doi.org/10.1145/2512349.2512804>
- Echtler et al.**, Supporting Casual Interactions between Board Games on Public Tabletop Displays and Mobile Devices, P9, <http://dx.doi.org/10.1007/s00779-009-0246-3>
- Fei et al.**, Positioning Portals with Peripheral NFC Tags to Embody Trans-Surface Interaction, P81, <http://dx.doi.org/10.1145/2512349.2514593>
- Forlines et al.**, Multi-User, Multi-Display Interaction with a Single-User, Single-Display Geospatial Application, P80, <http://dx.doi.org/10.1145/1166253.1166296>
- Forlines, Lilien**, Adapting a Single-user, Single-display Molecular Visualization Application for Use in a Multi-user, Multi-Display Environment, P10, <http://dx.doi.org/10.1145/1385569.1385635>
- Fukazawa et al.**, Comparison of Multimodal Interactions in Perspective-corrected Multi-display Environment, P74, <http://dx.doi.org/10.1109/3DUI.2010.5444711>
- Gjerlufsen et al.**, Shared Substance - Developing Flexible Multi-Surface Applications, P11, <http://dx.doi.org/10.1145/1978942.1979446>
- Hartmann et al.**, HydraScope - Creating Multi-Surface Meta-Applications Through View Synchronization and Input Multiplexing, P12, <http://dx.doi.org/10.1145/2491568.2491578>
- Heilig et al.**, Search, Explore and Navigate - Designing a Next Generation Knowledge Media Workbench, P88.
- Houben et al.**, ActivitySpace - Managing Device Ecologies in an Activity-Centric Configuration Space, P66, <http://dx.doi.org/10.1145/2669485.2669493>
- Jaing et al.**, System Design for the WeSpace - Linking Personal Devices to a Table-Centered Multi-User, Multi-Surface Environment, P71, <http://dx.doi.org/10.1109/TABLETOP.2008.4660191>
- Jetter et al.**, Model-Based Design and Implementation of Interactive Spaces for Information Interaction, P70, http://dx.doi.org/10.1007/978-3-642-16488-0_3
- Johanson et al.**, PointRight - Experience with Flexible Input Redirection in Interactive Workspaces, P89, <http://dx.doi.org/10.1145/571985.572019>
- Kaufmann et al.**, 3MF - A Service-Oriented Mobile Multimodal Interaction Framework, P53.
- Kim et al.**, Hugin - A Framework for Awareness and Coordination in Mixed-Presence Collaborative Information Visualization, P59, <http://dx.doi.org/10.1145/1936652.1936694>
- Kitamura et al.**, Multi-modal Interface in Multi-Display Environment for Multi-users, P69, http://dx.doi.org/10.1007/978-3-642-02577-8_8
- Kurdyukova et al.**, Studying User-defined iPad Gestures for Interaction in Multi-display Environment, P87, <http://dx.doi.org/10.1145/2166966.2166984>
- Lapides et al.**, Seamless Mixed Reality Tracking in Tabletop Reservoir Engineering Interaction, P75, <http://dx.doi.org/10.1145/2254556.2254694>
- Lee et al.**, Gesture-Based Interface for Connection and Control of Multi-device in a Tabletop Display Environment, P13, http://dx.doi.org/10.1007/978-3-642-02577-8_24
- Lee et al.**, Dual Interactions between Multi-display and Smartphone for Collaborative Design and Sharing, P72, <http://dx.doi.org/10.1109/VR.2011.5759478>
- Lee, Kim, Kim**, Gesture-Based Interactions on Multiple Large Displays with a Tabletop Interface, P37, http://dx.doi.org/10.1007/978-3-540-73281-5_102
- Luff et al.**, Flexible Ecologies and Incongruent Locations, P90, <http://dx.doi.org/10.1145/2702123.2702286>
- Maldonado et al.**, An Interactive Teachers Dashboard for Monitoring Groups in a Multi-tabletop Learning Environment, P68, http://dx.doi.org/10.1007/978-3-642-30950-2_62
- Maldonado et al.**, Multi-touch Technology in a Higher-Education Classroom - Lessons In-The-Wild, P85, <http://dx.doi.org/10.1145/2686612.2686647>
- McGrath et al.**, Branch-Explore-Merge - Facilitating Real-Time Revision Control in Co-Located Collaborative Visual Exploration, P38, <http://dx.doi.org/10.1145/2396636.2396673>
- Meissner, Eck**, Extending the Fire Dispatch System into the Mobile Domain, P14, http://dx.doi.org/10.1007/978-3-540-75668-2_16

- Nacenta et al.**, There and Back Again - Cross-Display Object Movement in Multi-Display Environments, P28, <http://dx.doi.org/10.1080/07370020902819882>
- Nacenta et al.**, A Comparison of Techniques for Multi-Display Reaching, P39, <http://dx.doi.org/10.1145/1054972.1055024>
- Nacenta et al.**, E-conic - a Perspective-Aware Interface for Multi-Display Environments, P40, <http://dx.doi.org/10.1145/1294211.1294260>
- Nacenta et al.**, Perspective Cursor - Perspective-Based Interaction For Multi-Display Environments, P64, <http://dx.doi.org/10.1145/1124772.1124817>
- Nacenta et al.**, The LunchTable - A Multi-User, Multi-Display System for Information Sharing in Casual Group Interactions, P79, <http://dx.doi.org/10.1145/2307798.2307816>
- Olwal, Feiner**, Spatially Aware Handhelds for High-Precision Tangible Interaction with Large Displays, P63, <http://dx.doi.org/10.1145/1517664.1517705>
- Oskamp et al.**, TerraGuide - Design and Evaluation of a Multi-Surface Environment for Terrain Visibility Analysis, P84, <http://dx.doi.org/10.1145/2702123.2702480>
- Peinsipp-Byma et al.**, Situation Analysis At A Digital Situation Table With Fovea-Tablett, P44, <http://dx.doi.org/10.1117/12.704367>
- Peinsipp-Byma, Geisler, Bader**, Digital Map and Situation Surface - A Team-oriented Multi-Display Workspace for Network Enabled Situation Analysis, P15, <http://dx.doi.org/10.1117/12.817903>
- Piazza et al.**, Holy Smartphones and Tablets, Batman - Mobile Interactions Dynamic Duo, P41, <http://dx.doi.org/10.1145/2525194.2525205>
- Ponnekanti et al.**, Portability, Extensibility and Robustness in iROS, P32, <http://dx.doi.org/10.1109/PERCOM.2003.1192722acmid=826373>
- Pratte et al.**, Exploring Multi-Surface Interactions in Retail Environments, P16, <http://dx.doi.org/10.1145/2669485.2669552>
- Prieto**, Studying Teacher Cognitive Load in Multi-tabletop Classrooms Using Mobile Eye-tracking, P52, <http://dx.doi.org/10.1145/2669485.2669543>
- Reilly et al.**, Mapping out Work in a Mixed Reality Project Room, P91, <http://dx.doi.org/10.1145/2702123.2702506>
- Rodrigues**, Bancada - Using Mobile Zoomable Lenses for Geospatial Exploration, P56, <http://dx.doi.org/10.1145/2669485.2669555>
- Roman et al.**, Gaia - A Middleware Infrastructure to Enable Active Spaces, P33, <http://dx.doi.org/10.1109/MPRV.2002.1158281>
- Sakurai et al.**, A Middleware for Seamless Use of Multiple Displays, P34, http://dx.doi.org/10.1007/978-3-540-70569-7_23
- Schmidt et al.**, A Cross-Device Interaction Style for Mobiles and Surfaces, P54, <http://dx.doi.org/10.1145/2317956.2318005>
- Schmidt et al.**, Personal Clipboards for Individual Copy-and-Paste on Shared Multi-User Surfaces, P60, <http://dx.doi.org/10.1145/2470654.2466457>
- Schmidt et al.**, PhoneTouch a technique for direct phone interaction on surfaces, P62, <http://dx.doi.org/10.1145/1866029.1866034>
- Schneider et al.**, WALDEN - Multi-Surface Multi-Touch Simulation of Climate Change and Species Loss in thoreaus Woods, P18, <http://dx.doi.org/10.1145/2396636.2396707>
- Scott et al.**, Surface Ghosts - Promoting Awareness of Transferred Objects during Pick-and-Drop Transfer in Multi-Surface Environments, P19, <http://dx.doi.org/10.1145/2669485.2669508>
- Scott et al.**, Cross-Device Transfer in a Collaborative MultiSurface Environment without User Identification, P73, <http://dx.doi.org/10.1109/CTS.2014.6867568>
- Seifert et al.**, MobiSurf - Improving Co-located Collaboration through Integrating Mobile Devices and Interactive Surfaces, P20, <http://dx.doi.org/10.1145/2396636.2396644>
- Seifert et al.**, From the private into the public - privacy-respecting mobile interaction techniques for sharing data on surfaces, P78, <http://dx.doi.org/10.1007/s00779-013-0667-x>

- Seyed et al.**, SkyHunter - A Multi-Surface Environment for Supporting Oil and Gas Exploration, P21, <http://dx.doi.org/10.1145/2512349.2512798>
- Seyed et al.**, Eliciting Usable Gestures for Multi-Display Environments, P45, <http://dx.doi.org/10.1145/2396636.2396643>
- Shen et al.**, Three Modes of Multi-Surface Interaction and Visualization, P46.
- Song**, WYSIWYF - Exploring and Annotating Volume Data with a Tangible Handheld Device, P58, <http://dx.doi.org/10.1145/1978942.1979140>
- Spindler**, Tangible Views for Information Visualization, P57, <http://dx.doi.org/10.1145/1936652.1936684>
- Spindler et al.**, Tangible Displays For The Masses - Spatial Interaction With Handheld Displays By Using Consumer Depth Cameras, P17, <http://dx.doi.org/10.1007/s00779-013-0730-7>
- Spindler, Bueschel, Dachsel**, Use Your Head - Tangible Windows for 3D Information Spaces in a Tabletop Environment, P47, <http://dx.doi.org/10.1145/2396636.2396674>
- Spindler, Stellmach, Dachsel**, PaperLens Advanced Magic Lens Interaction Above the Tabletop, P48, <http://dx.doi.org/10.1145/1731903.1731920>
- Streitz et al.**, i-LAND - An interactive Landscape for Creativity and Innovation, P49, <http://dx.doi.org/10.1145/302979.303010>
- Streitz et al.**, Roomware - Towards the next generation of human-computer interaction based on an integrated design of real and virtual worlds, P61.
- Tandler**, Architecture of BEACH - The Software Infrastructure for Roomware Environments, P35.
- Wagner et al.**, Body-centric Design Space for Multi-surface Interaction, P50, <http://dx.doi.org/10.1145/2470654.2466170>
- Wigdor et al.**, WeSpace - The Design, Development, and Deployment of a Walk-Up and Share Multi-Surface Collaboration System, P36, <http://dx.doi.org/10.1145/1518701.1518886>
- Wigdor et al.**, Table-Centric Interactive Spaces for Real-Time Collaboration, P51, <http://dx.doi.org/10.1145/1133265.1133286>
- Wigdor et al.**, Perception of Elementary Graphical Elements in Tabletop and Multi-Surface Environments, P83, <http://dx.doi.org/10.1145/1240624.1240701>
- Yao et al.**, Towards a Collaborative Urban Planning Environment, P22, http://dx.doi.org/10.1007/11686699_56
- Yoshimoto et al.**, Interaction Design of 2D-3D Map Navigation On Wall And Tabletop Displays, P23, <http://dx.doi.org/10.1145/2076354.2076402>

Glossary and Acronyms

- ABS** Acrylonitrile Butadiene Styrene. 125
- ACM** Association for Computing Machinery. 149
- API** Application Programming Interface. 128, 173–175, 177, 178, 182, 184–187, 194, 198, 199, 202, 206–210
- ARC** Automatic Reference Counting. 208
- AVI** Advanced Visual Interfaces. 149
- BPM** Beats per Minute. 101
- CDOM** Cross-Device Object Movement. 150, 151, 154, 155, 158
- Channel** A communication channel in multimodal interaction. 84
- CHI** Conference on Human Factors in Computing Systems. 149
- CLI** Common Language Infrastructure (.NET). 207
- Code** A code in multimodal interaction. 84
- COM** A serial communication port. 126, 127
- COM-objects** Component Object Model (Microsoft). 194
- CORBA** Common Object Request Broker Architecture. 158, 161
- CPU** Central Processing Unit. 186, 199, 202, 203
- CSCW** Computer Supported Collaborative Work. 142
- Design Space** A concept to frame possibilities by several dimensions. 15
- DHCP** Dynamic Host Configuration Protocol. 180
- DI** Diffuse Illumination. xii, 6, 23–26, 30, 31, 35, 45, 46, 49, 50, 52, 57, 62, 73, 160, 217
- DIY** Do-It-Yourself. 2
- DMCS** Display-Based Measurement and Control System. 118
- DOF** Degree Of Freedom. 154
- DPI** Dots Per Inch. 193
- DSI** Diffuse Surface Illumination. 23, 31, 35, 50
- ECG** Electrocardiography. 83, 90, 101
- EDA** Electrodermal Activity. 90, 101
- EEG** Electroencephalography. 83, 90, 101
- EICS** Engineering Interactive Computing Systems. 10, 149, 163
- EMG** Electromyography. 90, 101, 102
- FIFO** First-In First-Out. 186
- FTIR** Frustrated Total Internal Reflection. xii, 6, 23–26, 28, 29, 31, 35, 36, 45, 46, 49, 62, 65, 150, 217, 219
- FUBI** Full Body Interaction Framework. 133
- GPS** Global Positioning System. 194, 197
- GPU** Graphics Processing Unit. 198, 200, 202, 203
- GSR** Galvanic Skin Response. 90, 101
- GUI** Graphical User Interface. 167
- HCI** Human Computer Interaction. 15, 18, 43, 98, 99, 101, 148, 165, 167
- HEVC** High Efficiency Video Coding. 203
- HTML** Hypertext Markup Language. 171
- IEEE** Institute of Electrical and Electronics Engineers. 149, 160
- IoT** Internet of Things. 174, 208
- IP** Internet Protocol. 180
- ITS** Interactive Tabletops and Surfaces. 2, 10, 62, 81, 149, 163
- IUI** Intelligent User Interfaces. 149
- JNI** Java Native Interface. 188

- JPEG** Joint Photographic Experts Group. 199, 201, 203
- KCM** Kinematic Chain Model. 98, 99
- LAN** Local Area Network. 172, 189
- LCD** Liquid Crystal Display. 144
- LED** Light Emitting Diode. 28, 113
- LLP** Laser Light Plane. 23, 30
- MDE** Multi-Display Environment. 143, 145–148
- Medium** Medium in multimodal interaction. 84
- MERL** Mitsubishi Electric Research Laboratories. xii, 24, 37
- MLLP** Multi-Layer Laser Light Plane. 50
- Modality** A human perceivable or system measurable modality in multimodal interaction. 83
- Mode** Mode in multimodal interaction. 83
- MSE** Multi-Surface Environment. 7, 143, 145–150, 152, 153, 155, 157–160, 162–165, 167, 168, 170–172, 177–182, 188, 190, 193, 194, 211, 213, 222
- MVC** Model View Controller. 208
- NAT** Network Address Translation. 189, 190
- NFC** Near Field Communication. 152
- OMG** Object Management Group. 161
- PC** Personal Computer. 139
- PDA** Personal Digital Assistant. 38
- PerDis** Pervasive Displays. 149
- Phicons** Physical Icons. 105
- PIN** Personal Identification Number. 43
- PLA** Polylactic Acid. 125
- PNG** Portable Network Graphics. 201, 203
- PWM** Pulse Width Modulation. 127
- R/C cars** Remote controlled cars. 119
- RAM** Random Access Memory. 173, 174
- SDK** Software Development Kit. 3, 128
- SoC** System on a Chip. 203
- STUN** Session Traversal Utilities for NAT. 190, 191
- TAO** Tangible Active Objects. 116
- TCP** Transmission Control Protocol. 190–192, 194, 203
- TEI** Tangible and Embedded Interaction. 10, 117
- TTI** Tabletop Tangible Interface. 103
- TUI** Tangible User Interface. 103, 106
- TUIO** Tangible User Interface Objects. 41, 195
- UbiComp** Pervasive and Ubiquitous Computing. 149
- UDP** User Datagram Protocol. 190–195
- UI** User Interface. 150
- UIST** User Interface Software and Technology. 149
- USB** Universal Serial Bus. 141
- Visual Cortex** Human visual cortex V5/MT for motion perception. 115
- VNC** Virtual Network Computing. 166, 201
- VOC** Volatile Organic Compound. 197
- WiFi** Wireless Fidelity. 172
- WIMP** Windows, Icons, Menus, and Pointers. 20, 39
- WPF** Windows Presentation Foundation. 162, 201, 204
- XML** Extensible Markup Language. 59, 73, 133
- ZOIL** Zoomable Object-Oriented Information Landscape. 148, 162